Can Artificial Intelligence Discover Mathematics Before Humans Name It? Automated Theory Formation, Machine-Generated Mathematics, and Algebraic Abiogenesis

Douglas C. Youvan

doug@youvan.com www.youvan.ai

July 13, 2026

Artificial intelligence has moved from checking proofs to searching for proofs, proposing conjectures, inventing concepts, and, increasingly, organizing mathematical theories. Yet most celebrated systems still begin with a named domain, a supplied vocabulary, a target theorem, or an externally defined objective. This review reconstructs the development of machine-generated mathematics from early automated discovery programs such as AM and HR through conjecture engines, automated theorem provers, finite model builders, program-synthesis systems, formal proof assistants, and contemporary agentic co-mathematicians. It then identifies a deeper frontier: whether a machine can discover mathematical structure before humans specify the objects, axioms, or theories to be found. Algebraic Abiogenesis is examined as an experimental response to that question. Beginning with finite carriers and anonymous operations, it generates expressions, detects exact term-function collisions, proposes candidate laws, rejects overgeneralizations through countermodels, and allows theories to emerge before they are named. The paper argues that this operation-first approach extends automated theory formation beneath conventional mathematical ontology, while also clarifying its precedents, limitations, evaluation criteria, and requirements for reproducible, human-governed discovery. Its novelty must therefore be evaluated historically, experimentally, and cautiously.

Keywords: artificial intelligence for mathematics, automated theory formation, machine-generated mathematics, mathematical discovery, automated theorem proving, conjecture generation, concept invention, ontology formation, algebraic abiogenesis, abstract algebra, universal algebra, finite algebra, anonymous operations, equational theory, term functions, countermodel generation, formal verification, human-AI collaboration, agentic mathematics, neurosymbolic reasoning.

A collaboration with OpenAI GPT-5.5 Thinking. CC4.0. 90 pages and 59 references.

https://orcid.org/0009-0004-4206-4253

Outline

1. From Mechanical Calculation to Machine-Generated Mathematics

1.1 Calculation, deduction, and discovery

1.2 The distinction between verifying and originating mathematics

1.3 Mathematical creativity as a computational research problem

1.4 Five levels of machine participation in mathematics

1.5 The unresolved problem of autonomous theory formation

2. Early Programs for Automated Mathematical Discovery

2.1 Logic machines and the automation of deduction

2.2 The Logic Theorist and General Problem Solver

2.3 AM and heuristic concept formation

2.4 Eurisko and the recursive modification of discovery rules

2.5 Lessons and limitations of early symbolic artificial intelligence

3. HR and the Classical Program of Automated Theory Formation

3.1 Concept invention as a computational operation

3.2 Production rules, examples, classifications, and conjectures

3.3 The integration of theorem provers and model generators

3.4 Machine-generated integer sequences and mathematical novelty

3.5 Why HR is the principal historical precursor to Algebraic Abiogenesis

3.6 The ontological assumptions retained by HR

4. Conjecture Engines and Data-Guided Mathematics

4.1 Graffiti and automated inequality generation

4.2 TxGraffiti and the filtering of conjectural abundance

4.3 The Ramanujan Machine and formula discovery

4.4 Symbolic regression and the recovery of mathematical laws

4.5 Interestingness, surprise, and the selection problem

4.6 Why conjecture generation is not yet theory formation

5. Automated Theorem Proving, Countermodels, and Formal Proof

5.1 Resolution, rewriting, and first-order theorem proving

5.2 Prover9, Vampire, E, and related proof-search systems

5.3 Mace4 and finite countermodel generation

5.4 Coq, Agda, Isabelle, HOL, and Lean

5.5 The computer-assisted development of Homotopy Type Theory

5.6 From human-written proof scripts to AI-generated formalization

6. Machine Learning and the New Mathematics of Search

6.1 Neural guidance for symbolic proof search

6.2 AlphaZero-style reasoning and reinforcement learning

6.3 AlphaGeometry and hybrid symbolic-neural deduction

6.4 AlphaProof and formal mathematical competition solving

6.5 FunSearch, AlphaTensor, and AlphaEvolve

6.6 Discovery under a supplied evaluator

6.7 The difference between optimization and open-ended mathematics

7. Agentic Co-Mathematicians and Human-Governed Discovery

7.1 Large language models as mathematical research agents

7.2 Tool use, computation, literature search, and persistent memory

7.3 Multi-agent criticism and hypothesis revision

7.4 Human strategic intervention in machine-led exploration

7.5 AI-assisted concept recovery and the rediscovery of homology

7.6 FERMAT and environments for open-ended theory construction

7.7 The AI co-mathematician as a research-workflow architecture

8. Equational Reasoning and Computational Abstract Algebra

8.1 Universal algebra as a natural domain for machine discovery

8.2 Terms, identities, varieties, and finite models

8.3 Automated reasoning in quasigroups, loops, magmas, and lattices

8.4 The Equational Theories Project

8.5 Enumerated laws, implication graphs, and counterexamples

8.6 The boundary between mapping a theory and generating one

9. Algebraic Abiogenesis: Mathematics Before Names and Axioms

9.1 The operation-first starting point

9.2 Finite carriers and anonymous operation tables

9.3 Generation of formal expressions

9.4 Exact evaluation and term-function signatures

9.5 Collisions as candidate identities

9.6 Countermodels and the rejection of accidental regularities

9.7 Recursive theory peeling and theory ecologies

9.8 The emergence of objects, laws, and algebraic interpretation

10. Comparative Analysis and the Question of Novelty

10.1 Algebraic Abiogenesis and AM

10.2 Algebraic Abiogenesis and HR

10.3 Algebraic Abiogenesis and conjecture generators

10.4 Algebraic Abiogenesis and automated theorem provers

10.5 Algebraic Abiogenesis and FERMAT

10.6 Algebraic Abiogenesis and the Equational Theories Project

10.7 Algebraic Abiogenesis and self-supervised theorem discovery

10.8 The distinctive conjunction of anonymity, exactness, and target freedom

10.9 Claims that can and cannot presently be defended

11. Evaluation, Certification, and Reproducibility

11.1 What counts as a machine-discovered mathematical theory

11.2 Rediscovery, novelty, and historical comparison

11.3 Syntactic novelty versus semantic novelty

11.4 Compression, explanatory power, and theorem productivity

11.5 Robustness across finite operations and carrier sizes

11.6 Countermodel resistance and generalization

11.7 Formal certification and proof-assistant integration

11.8 GitHub repositories as executable scientific archives

11.9 Human oversight, provenance, and intellectual accountability

12. Toward an Autonomous Ecology of Mathematical Discovery

12.1 From theorem proving to ontology formation

12.2 Human nonlinear insight and AI linear execution

12.3 The preservation of separation between mathematician and machine

12.4 Open-ended theory populations and evolutionary search

12.5 Machine-learned mathematical interestingness

12.6 Cross-domain extension beyond finite algebra

12.7 Risks of triviality, rediscovery, and hidden human targets

12.8 A research agenda for post-axiomatic mathematical discovery

12.9 Can mathematics be born before it is named?

13. References

1. From Mechanical Calculation to Machine-Generated Mathematics

The history of computational mathematics is often told as a steady progression from faster arithmetic to more powerful theorem proving. That account is incomplete. Calculation, deduction, conjecture formation, concept invention, and theory construction are distinct activities, and progress in one does not automatically imply progress in the others. A machine that evaluates a million expressions per second may contribute nothing to the discovery of a new mathematical object. A theorem prover may establish a difficult result without understanding why the theorem is interesting. A language model may generate plausible mathematical prose without producing a valid argument. The central question is therefore not whether computers can perform mathematical operations. They plainly can. The deeper question is whether they can participate in the formation of mathematics itself: selecting patterns, inventing useful distinctions, proposing laws, organizing results, and producing theories that were not already implicit in a human-specified target.

This distinction matters because mathematics is not merely a collection of true propositions. It is also a historical process by which objects, languages, axioms, and standards of importance are created. Groups, rings, manifolds, categories, and topological spaces were not simply waiting as named entities inside a fixed formal universe. They emerged when mathematicians recognized that disparate problems shared common structural features. The creative act often consisted not in proving one more theorem about familiar objects, but in deciding what the objects should be. Any serious account of machine-generated mathematics must therefore address not only automated reasoning within established theories but also the possibility of automated theory formation.

1.1 Calculation, Deduction, and Discovery

Calculation applies known operations to supplied inputs. Deduction derives consequences from supplied assumptions according to accepted rules. Discovery introduces something not previously specified as the immediate goal: a conjecture, concept, invariant, construction, classification, proof strategy, or theory. These activities overlap, but they should not be conflated.

A computer algebra system may simplify an expression, factor a polynomial, integrate a function, or solve a system of equations. These achievements can be indispensable to research, yet the mathematical task has already been framed. The objects, operations, and desired output are known. The machine performs an extended calculation that would otherwise burden a human investigator.

Automated deduction begins at a higher logical level. Given axioms and a target proposition, a theorem prover searches for a derivation. The resulting proof may be surprising, shorter than

known proofs, or beyond unaided human reach. Nevertheless, the problem ordinarily arrives pre-formed. Humans determine the language, axioms, definitions, and theorem to be proved. The computer searches inside a mathematical world whose ontology has already been supplied.

Discovery changes the specification itself. It may reveal that a previously unnoticed quantity is invariant, that two apparently separate domains share a structure, that a failed conjecture becomes true under a refined hypothesis, or that a family of examples should be recognized as a new kind of object. Discovery may therefore alter the space in which subsequent deductions occur. It does not merely traverse a theory. It may reorganize or enlarge the theory.

The strongest form of machine-generated mathematics would combine all three activities. It would calculate examples, deduce consequences, and use those results to revise its own concepts and questions. It would move recursively between data, language, law, and proof.

1.2 The Distinction Between Verifying and Originating Mathematics

Verification asks whether a proposed claim follows from stated assumptions. Origination asks where the claim, assumptions, and conceptual vocabulary came from. This distinction separates most traditional computer-assisted mathematics from the more ambitious project of artificial mathematical discovery.

Proof assistants provide exceptionally strong verification. They require mathematical arguments to be expressed in a formal language and checked against precise logical rules. When a proof is accepted, the result has a level of syntactic and inferential reliability that ordinary prose rarely provides. Yet a proof assistant does not usually decide which theorem should be stated. It does not independently determine that a new definition captures an important phenomenon. It does not ordinarily recognize that a technical lemma should become the foundation of a new field.

Automated theorem provers move closer to origination by searching for proofs rather than merely checking human-written proof terms. Even so, most operate under a supplied goal. Their creativity is local: they originate a route through an established logical space. They do not necessarily originate the space itself.

A machine begins to originate mathematics when it participates in selecting what is worth proving, inventing the concepts needed to express it, and reorganizing results into a coherent body of theory. Such origination need not require consciousness, intention, or human-like understanding. It can be defined operationally. A system originates mathematical content when its output is not merely a direct execution of a fully specified human plan and when the output introduces a nontrivial structure, statement, or explanatory organization that was not explicitly encoded in advance.

This criterion remains difficult to apply. Every computational system inherits representations, algorithms, objectives, and evaluation rules from human designers. The question is therefore not whether human influence can be eliminated. It is whether the decisive mathematical content was supplied directly or emerged from a sufficiently open process.

1.3 Mathematical Creativity as a Computational Research Problem

Mathematical creativity is sometimes treated as too mysterious for formal investigation. That position prevents useful distinctions. Creativity can be decomposed into partially observable operations: generating candidates, transforming representations, detecting analogies, measuring novelty, searching for counterexamples, compressing regularities, introducing definitions, and evaluating explanatory power.

Each component can become a computational research problem. Candidate generation can be studied through symbolic enumeration, stochastic search, program synthesis, or language models. Analogy can be approached through structural mappings between domains. Novelty can be estimated against databases of known results. Interestingness can be approximated through compression, unexpected implication, reuse across proofs, explanatory unification, or expert judgment. Counterexample search can be automated over finite models. Definitions can be evaluated by the theorems they enable and the complexity they remove.

No individual measure captures mathematical creativity. A rare statement may be useless. A highly compressive law may be trivial. A prolific definition may merely repackage known material. A system may rediscover a famous theorem without historical awareness and mistake rediscovery for novelty. The computational challenge is therefore not simply to maximize one score but to coordinate multiple imperfect criteria.

Mathematical creativity also depends on scale. Automated systems can generate enormous numbers of true statements. This abundance creates a selection crisis. Human mathematicians do not value mathematics merely because it is correct. They seek results that explain, connect, simplify, generalize, or open productive lines of inquiry. The future of machine-generated mathematics will depend less on producing additional formal truths than on constructing mechanisms that distinguish fertile mathematics from combinatorial debris.

1.4 Five Levels of Machine Participation in Mathematics

Machine participation can be organized into five progressively stronger levels.

The first level is verification. The machine checks arithmetic, symbolic transformations, proofs, or formal derivations supplied by humans. It functions as a reliability instrument.

The second level is proof discovery. The language, assumptions, and theorem are supplied, but the machine finds some or all of the argument. It functions as a search engine inside a defined theory.

The third level is conjecture discovery. The machine examines examples, computations, or databases and proposes statements not explicitly requested in advance. It begins to influence the research agenda, although the object domain and descriptive vocabulary are usually predetermined.

The fourth level is concept invention. The machine introduces new predicates, invariants, classifications, constructions, or representations that make previously inaccessible regularities expressible. It changes not only what can be proved but what can be stated.

The fifth level is theory or ontology formation. The machine organizes concepts and laws into a coherent mathematical world, potentially identifying its fundamental objects, equivalences, transformations, and explanatory principles. At this level, the system does not merely solve problems within a domain. It participates in determining what the domain is.

These levels are not perfectly separated. A proof-discovery system may introduce a useful lemma that functions as a new concept. A conjecture generator may reorganize a field by revealing an unexpected invariant. A theory-forming system still depends on verification and deduction. The hierarchy is therefore functional rather than absolute. Its purpose is to prevent impressive achievements at one level from being mistaken for achievements at another.

1.5 The Unresolved Problem of Autonomous Theory Formation

Autonomous theory formation remains the least developed level. Existing systems can generate proofs, equations, programs, conjectures, and classifications. Some can modify their own search strategies or invent derived concepts. Yet most begin with a substantial inherited ontology. They are told that the relevant objects are numbers, graphs, groups, geometric configurations, programs, or formal propositions. They are given predicates, constructors, axioms, evaluators, training corpora, or benchmark goals that already encode much of the mathematical world they will explore.

The unresolved problem is whether a computational system can begin beneath this level of specification. Can it encounter minimally interpreted structure and discover stable objects, useful equivalences, candidate laws, and organizing principles without being told which familiar theory to reconstruct? Can it distinguish genuine structure from accidental regularity? Can it form multiple competing theories, compare their explanatory productivity, reject weak generalizations, and preserve the strongest survivors? Can it recognize when a new vocabulary is required?

No system begins from literal nothing. A carrier set, representation scheme, computational substrate, or observation policy must always be supplied. The meaningful question is how little mathematical interpretation must be imposed before theory formation begins. The boundary between necessary infrastructure and covertly supplied mathematics is central.

Algebraic Abiogenesis enters at this boundary. It does not attempt to eliminate all prior structure. Instead, it reduces the initial commitment to finite carriers, anonymous operations, generated expressions, and exact evaluation. It then asks whether identities, theory families, and eventually recognizable algebraic objects can emerge from repeated behavioral equivalence. The project therefore belongs to a longer history of automated discovery while advancing a more specific challenge: whether mathematics can be induced before its objects and axioms have been named.

2. Early Programs for Automated Mathematical Discovery

The ambition to mechanize mathematical reasoning predates digital computers. Early calculating devices showed that arithmetic procedures could be externalized, while formal logic suggested that at least some reasoning might be reduced to rule-governed symbol manipulation. By the middle of the twentieth century, these strands converged in the emerging fields of computer science and artificial intelligence. Researchers no longer asked only whether machines could calculate. They asked whether machines could prove theorems, solve unfamiliar problems, discover useful concepts, and reproduce parts of scientific creativity.

The early systems were constrained by limited memory, slow processors, and brittle symbolic representations. Nevertheless, they established nearly every major question that continues to shape artificial intelligence for mathematics. How should mathematical knowledge be represented? Which search steps should be attempted first? Can a machine distinguish a promising conjecture from a meaningless one? Should discovery be guided by fixed heuristics, learned strategies, or recursive self-modification? How much domain knowledge must be encoded before genuine discovery becomes possible?

These systems did not achieve autonomous mathematical science. Their importance lies elsewhere. They demonstrated that mathematical activity could be decomposed into computational operations and that machines could sometimes produce results not explicitly written into their programs. They also exposed a persistent difficulty: the more intelligence designers encoded into the initial representation and heuristics, the harder it became to determine whether the machine had discovered mathematics or merely unfolded the mathematics already placed inside it.

2.1 Logic Machines and the Automation of Deduction

The first major route toward machine mathematics came through formal logic. Nineteenth- and early twentieth-century developments had shown that substantial portions of mathematics could be represented using symbolic languages, axioms, and explicit rules of inference. Once reasoning was expressed as symbol manipulation, the possibility of mechanization became unavoidable.

The attraction was clear. Informal mathematical arguments depend on interpretation, convention, and unspoken background knowledge. Formal deduction replaces these with precisely specified transformations. A machine need not understand numbers, sets, or geometry in a human sense. It needs only to determine whether one symbolic expression can be transformed into another under permitted rules.

Early logical machines and formal reasoning procedures therefore concentrated on decision problems, satisfiability, proof search, and the derivation of consequences from axioms. The underlying conception of mathematics was strongly deductive. A theory consisted of primitive symbols, axioms, and inference rules. Mathematical work consisted of deriving theorems.

This conception produced foundational advances, but it also narrowed the meaning of discovery. A deduction system could establish a theorem only after humans had selected the language, formalized the axioms, and specified the desired result. It could reveal an unexpected proof path, but it did not necessarily decide which theorem mattered or which concepts should exist.

The limitation became more visible as automated reasoning improved. Search spaces grew explosively. Even when a proof was logically obtainable, blindly applying inference rules generated vast numbers of irrelevant expressions. Mathematical reasoning therefore required more than correctness. It required strategy.

This recognition introduced a central theme of artificial intelligence: search must be guided. The machine must choose among possible deductions, prioritize promising paths, abandon unproductive branches, and reuse intermediate results. These strategic decisions resemble aspects of mathematical judgment, even when they remain encoded as explicit procedures.

2.2 The Logic Theorist and General Problem Solver

The Logic Theorist, created in the 1950s by Allen Newell, Herbert Simon, and Cliff Shaw, is widely regarded as one of the first artificial intelligence programs. It was designed to prove theorems from propositional logic, particularly results appearing in a standard foundational text. Rather than exhaustively enumerate every possible derivation, it used heuristics intended to imitate selective human reasoning.

The system represented proof as a search through possible symbolic transformations. It worked backward from a desired theorem, identified differences between the current expression and the goal, and selected operations likely to reduce those differences. This approach was significant because it treated theorem proving not merely as formal deduction but as intelligent problem solving.

The Logic Theorist successfully proved many theorems and reportedly found at least one proof regarded as more elegant than the published version. That result became an early symbol of machine creativity. The program had not merely verified a human proof. It had produced an alternative derivation through its own search.

Yet the achievement remained bounded. The target theorems were supplied. The logical language was fixed. The inference rules were known. The machine's originality lay in navigating a predetermined formal space.

Newell and Simon generalized the underlying method in the General Problem Solver. The goal was to construct a domain-independent architecture capable of solving problems by comparing an initial state with a desired state and applying operators that reduced the difference. This method, often called means–ends analysis, became influential across artificial intelligence.

The General Problem Solver expressed an enduring hope: that reasoning might be separated into a general search architecture and a domain-specific description of states and operators. If successful, one computational mechanism could support mathematics, planning, puzzles, and scientific reasoning.

The difficulty was that real domains resist clean separation. The representation of a problem often determines whether it is easy or impossible to solve. Experts do not merely search more efficiently than novices. They perceive different objects, choose different abstractions, and reformulate problems. The General Problem Solver could operate within a representation but was less capable of inventing the representation itself.

This limitation foreshadowed the distinction between theorem proving and theory formation. Solving a stated problem is important, but mathematics frequently advances by changing the statement, redefining the objects, or discovering that the apparent problem belongs to a different conceptual domain.

2.3 AM and Heuristic Concept Formation

Douglas Lenat's AM system marked a more ambitious attempt to automate mathematical discovery. Developed during the 1970s, AM was intended not merely to prove supplied theorems but to explore elementary mathematics, invent concepts, generate conjectures, and evaluate the apparent interest of its findings.

AM began with a collection of basic concepts and hundreds of heuristic rules. These rules suggested possible transformations of existing concepts, encouraged exploration of unusual examples, and assigned measures of interest to newly generated structures. The system could define new concepts by specializing, generalizing, composing, or modifying those already available.

This was a major conceptual shift. Mathematical discovery was represented as movement through a space of possible concepts. A system did not need to wait for humans to state every theorem. It could generate new predicates, examine their instances, notice regularities, and formulate conjectures about them.

AM is often credited with rediscovering familiar mathematical ideas, including concepts related to prime numbers. Such episodes demonstrated that computational concept formation could reproduce recognizable mathematical structure from more elementary ingredients. They also raised a difficult question: was the system genuinely discovering, or were its encoded heuristics already shaped so strongly by mathematical knowledge that the resulting discoveries were effectively anticipated?

The answer is not binary. AM clearly performed nontrivial search and generated combinations not individually specified by its designers. At the same time, its success depended heavily on human-crafted representations and heuristics. Its rules embodied assumptions about what kinds of concepts mathematicians tend to find useful.

This dependence does not invalidate the system. Human mathematicians also inherit languages, examples, values, and disciplinary traditions. No discovery occurs without prior structure. The relevant question is whether the system's output could have been predicted directly from its instructions or whether the interaction of its rules generated mathematically meaningful results that were not explicitly encoded.

AM also confronted the problem of interestingness. A concept generator can produce enormous numbers of definitions. Most are formally valid but mathematically sterile. The system therefore needed heuristics that favored concepts with unusual examples, simple descriptions, connections to existing ideas, or productive conjectures.

This problem remains unresolved. Modern systems can generate vastly more mathematical material than AM, but the selection problem has only become more severe. The central challenge is not the production of novelty in the weak sense of unfamiliar symbolic combinations. It is the production of novelty that reorganizes understanding.

2.4 Eurisko and the Recursive Modification of Discovery Rules

Lenat extended the ideas of AM in Eurisko, a system designed to operate not only on domain concepts but also on its own heuristics. Eurisko could propose, modify, combine, and evaluate rules for discovery. This introduced a recursive level of search: the machine explored both a problem space and the strategies used to explore that space.

The motivation was compelling. A fixed collection of heuristics may work well in one domain and fail in another. If a system can alter its own methods, it may adapt to new environments and discover strategies not anticipated by its designers. In principle, recursive heuristic modification could allow computational creativity to become increasingly autonomous.

Eurisko was applied to several domains and became especially famous for generating successful strategies in a complex naval-design game. Its achievements were not purely mathematical, but they illustrated the power of systems that search over procedures rather than only over solutions.

For automated mathematics, the importance of Eurisko lies in its recognition that discovery rules are themselves objects of discovery. A system may need to learn when to generalize, when to search for counterexamples, when to introduce a new invariant, when to abandon a vocabulary, or when to divide one theory into several competing branches.

This recursive flexibility also creates instability. A system that modifies its heuristics can generate self-reinforcing but unproductive behaviors. It may exploit defects in evaluation criteria, favor trivial results that score well, or drift away from the intended domain. Human oversight and carefully designed feedback remain necessary.

Eurisko therefore anticipated both the promise and danger of modern agentic systems. Contemporary AI agents can revise plans, write tools, inspect failures, and modify their own workflows. Yet they may also optimize proxies, invent unsupported claims, or become trapped in repetitive loops. The core problem is not simply self-modification. It is self-modification governed by criteria aligned with genuine mathematical value.

2.5 Lessons and Limitations of Early Symbolic Artificial Intelligence

Early symbolic systems established that parts of mathematical discovery could be computationally represented. They proved that machines could search for proofs, apply heuristics, invent derived concepts, generate conjectures, and alter discovery strategies. These achievements were conceptually profound even when the resulting mathematics remained elementary.

They also exposed several limitations that continue to define the field.

The first was combinatorial explosion. Formal systems permit far more syntactically valid moves than useful ones. Without strong guidance, search becomes computationally impossible. Mathematical intelligence therefore requires mechanisms for prioritization, abstraction, and pruning.

The second was representation dependence. A system's apparent intelligence could change dramatically when the same problem was encoded differently. Good representations compress relevant structure and make useful operations accessible. Bad representations conceal relationships and produce unmanageable search spaces. Automated discovery must therefore address not only reasoning within representations but the invention and revision of representations.

The third was heuristic brittleness. Handwritten rules often captured the intuitions of their creators but failed outside the domains for which they were designed. Systems appeared creative when their heuristics aligned with the problem and incompetent when those assumptions no longer applied.

The fourth was the evaluation problem. A machine could generate true statements and novel concepts in abundance, but truth and novelty were insufficient. Mathematicians value depth, explanatory power, surprising connection, generality, elegance, and future productivity. These qualities are difficult to express as fixed computational objectives.

The fifth was hidden human contribution. The more domain knowledge designers supplied through predicates, operators, examples, and heuristics, the harder it became to isolate the machine's contribution. This remains a central issue in claims of autonomous discovery. A system may appear to invent a concept while operating inside a vocabulary designed to make that concept likely.

Despite these limitations, early symbolic artificial intelligence should not be dismissed as a failed prehistory superseded by machine learning. Its central questions remain current. Modern systems have more computation, larger knowledge bases, statistical learning, formal proof environments, and natural-language interfaces. Yet they still must search, represent, evaluate, and revise.

The early programs also supplied a crucial warning for Algebraic Abiogenesis. Reducing explicit prior knowledge does not automatically produce unbiased discovery. Bias can enter through the expression grammar, enumeration order, complexity measure, carrier size, operation sampling policy, equivalence criterion, or selection rule. The claim that mathematics emerges from anonymous operations must therefore be accompanied by a precise account of what remains supplied.

The enduring legacy of these systems is the recognition that mathematical discovery can be treated as an iterative computational ecology. Concepts generate examples. Examples generate conjectures. Conjectures provoke proofs and counterexamples. Failures alter concepts. Successful concepts reorganize the search. Later systems would make this cycle more explicit, more scalable, and more formally integrated. The most important bridge between the early symbolic era and contemporary Algebraic Abiogenesis would be the HR system and its program of automated theory formation.

3. HR and the Classical Program of Automated Theory Formation

Among the early systems for machine-generated mathematics, HR occupies a uniquely important position. It did not merely prove theorems inside a fixed theory, nor did it restrict itself to searching for isolated conjectures. It was designed to construct mathematical theories by inventing concepts, calculating examples, identifying regularities, formulating conjectures, seeking proofs, generating counterexamples, and evaluating the apparent interest of its own results. For that reason, HR is the principal historical precursor to contemporary attempts at open-ended mathematical discovery.

HR also forces a more disciplined assessment of novelty. Any modern claim that a machine can invent mathematical concepts, organize results, or construct a theory must be compared with what HR had already attempted decades earlier. At the same time, HR began with substantial mathematical structure already in place. It was supplied with recognizable domains, primitive concepts, relations, and production rules. Its achievement was therefore not mathematics from nothing, but theory formation within a partially interpreted mathematical world.

This distinction is central to Algebraic Abiogenesis. The novelty of the newer approach cannot rest on the general idea of automated theory formation, because HR already established that program. The stronger claim is that Algebraic Abiogenesis attempts to move the starting point below named mathematical domains by beginning with anonymous finite operations and allowing candidate laws to emerge from exact behavioral equivalence.

3.1 Concept Invention as a Computational Operation

HR treated concept invention as an explicit computational process. A mathematical concept was represented extensionally through the objects satisfying it and intensionally through a definition describing how it had been constructed. New concepts arose by applying production rules to previously available concepts.

This approach transformed definition-making from a mysterious act into a sequence of operations. A system could specialize a concept, generalize it, combine it with another, quantify

over parts of a structure, negate a condition, count instances, compose relations, or introduce auxiliary constructions. Each production rule generated a possible new way of classifying mathematical objects.

The resulting concepts were not always profound. Many were trivial, redundant, or awkward. Yet this was not fundamentally different from human exploration. Mathematicians also propose definitions that later prove sterile. The difficulty is not generating definitions but identifying those that reveal structure.

HR therefore evaluated concepts using measures intended to approximate mathematical interest. A concept might receive attention because it classified examples in a nontrivial way, connected previously separate ideas, supported new conjectures, or produced unexpected patterns. Concept invention and concept evaluation were inseparable.

The importance of this framework lies in its operational account of mathematical ontology. The objects of a theory are not merely given. They can be reorganized by new predicates and classifications. Once a useful concept is introduced, previously scattered examples become instances of a common structure, and new statements become expressible.

This is an essential step beyond theorem proving. A theorem prover reasons with concepts already available in its language. A theory-forming system can alter that language by introducing new concepts. It changes the space of possible theorems.

3.2 Production Rules, Examples, Classifications, and Conjectures

HR's discovery cycle began with a collection of primitive concepts and examples. It then applied production rules to generate new concepts, calculated which objects satisfied them, and compared the resulting classifications.

Examples played several roles. They grounded abstract concepts in concrete cases, exposed regularities, suggested conjectures, and supplied evidence against overgeneralization. A concept with an empty extension might be discarded. A concept coinciding exactly with another could suggest an equivalence. Two concepts whose extensions stood in an inclusion relation could suggest an implication.

From these extensional relationships, HR generated conjectures of several kinds. If two concepts classified exactly the same objects, the system could conjecture that they were equivalent. If every example satisfying one concept also satisfied another, it could conjecture an implication. If concepts divided a domain into apparently exhaustive classes, it could conjecture a classification theorem.

This method is important because it connects computation with formal statement generation. The machine does not invent conjectures from unconstrained linguistic association. It derives them from observed relationships among explicitly calculated examples.

The same general architecture appears in modern systems, although the computational tools are more powerful. A system generates candidate structures, evaluates them, detects patterns, forms conjectures, and submits those conjectures to proof or counterexample search. The intellectual continuity between HR and contemporary agentic mathematics is therefore substantial.

However, the method also risks overfitting finite data. Two concepts may agree on all examined examples while differing elsewhere. An implication may hold for small cases but fail at larger sizes. The system must therefore distinguish empirical coincidence from mathematical necessity.

This is where theorem provers and model generators become essential.

3.3 The Integration of Theorem Provers and Model Generators

HR did not treat conjecture generation as the end of discovery. It could interact with automated theorem provers to establish conjectures and with model generators to find counterexamples.

This integration created a more complete mathematical loop. Extensional data suggested a law. A theorem prover attempted to derive it from the domain axioms. If proof search failed, a model generator attempted to construct a structure in which the conjecture was false. A counterexample could then refine the concept or invalidate the proposed relationship.

The proof–countermodel duality is one of the most powerful architectures in computational mathematics. Proof establishes that a law follows generally from assumptions. A finite countermodel demonstrates that an apparent pattern is accidental or too broadly stated. Together, they prevent a discovery system from confusing repeated observation with theoremhood.

This architecture also supports theory revision. A failed conjecture need not be discarded without residue. Its counterexample may reveal a missing hypothesis. The system can introduce a more selective concept, partition the domain, or generate a corrected statement. Failure becomes a source of concept formation.

In human mathematics, counterexamples often have this creative role. They do not merely refute. They expose the boundary of a concept and motivate better definitions. An automated theory-forming system should therefore preserve counterexamples as structural information rather than treating them as terminal errors.

HR's use of external reasoning tools also anticipated contemporary tool-using AI agents. Modern systems may combine language models, proof assistants, symbolic algebra, satisfiability solvers, finite model finders, numerical experiments, databases, and custom code. HR demonstrated early that no single reasoning method is sufficient for broad mathematical discovery.

3.4 Machine-Generated Integer Sequences and Mathematical Novelty

One of HR's most frequently cited achievements was the generation of previously unrecorded integer sequences that were accepted into a recognized mathematical sequence database. This provided a concrete form of novelty. The outputs were not merely rediscoveries of standard textbook concepts. They were sufficiently distinct to be cataloged as new mathematical objects.

The significance of these sequences should be interpreted carefully. Producing a new sequence is easier than producing a deep theory. Infinitely many sequences can be generated by arbitrary definitions. Novelty alone does not establish mathematical importance.

HR's stronger contribution was that it attempted to explain why a generated sequence might be interesting. A sequence could arise from a concept connected to number-theoretic properties, unusual classifications, or conjectured relationships. The system therefore linked object generation with explanatory context.

This points to a general requirement for machine-generated mathematics. A discovery should not be judged only by whether it is absent from existing databases. It should also be assessed by whether it supports proofs, connects concepts, compresses patterns, generates further questions, or reveals a structure with broader use.

Historical novelty is also difficult to establish automatically. A system may generate a result not present in its local knowledge base but already known elsewhere under different terminology. Mathematical equivalence can be obscured by notation, presentation, or disciplinary boundaries. Claims of machine novelty therefore require literature search and expert comparison.

HR's sequence results nevertheless remain important because they demonstrated that computational theory formation could produce outputs not individually anticipated by the designers and recognizable as legitimate additions to mathematical knowledge.

3.5 Why HR Is the Principal Historical Precursor to Algebraic Abiogenesis

HR is the closest historical precursor to Algebraic Abiogenesis because both projects treat mathematics as an evolving ecology rather than a fixed set of theorem-proving tasks.

Both involve:

- the generation of candidate concepts or expressions;
- exact computation over finite examples;
- detection of regularities;
- formulation of conjectures;
- proof or counterexample search;
- ranking by mathematical interest;
- iterative refinement of a growing theory.

Both also reject the idea that the machine's role should be limited to checking a finished human argument. The machine participates earlier, during the formation of the mathematical questions and vocabulary.

The relationship is therefore not one of total discontinuity. Algebraic Abiogenesis belongs to the automated theory formation lineage established by HR. A credible account of the newer project should acknowledge this ancestry directly.

The difference lies in the initial ontology. HR normally begins with a recognizable mathematical domain. It may be given integers, graphs, groups, or other objects together with primitive predicates and relations. Its production rules then construct new concepts inside that domain.

Algebraic Abiogenesis attempts to begin with less semantic commitment. It starts with a finite carrier and one or more operation tables whose conventional mathematical interpretation is withheld. Expressions are built syntactically, evaluated exactly, and grouped by identical term-function behavior. Candidate laws arise from these collisions before the operation is classified as associative, commutative, idempotent, group-like, lattice-like, or anything else.

Thus HR asks:

What new concepts and conjectures can be invented inside a supplied mathematical domain?

Algebraic Abiogenesis asks:

What algebraic domain can emerge when the operation is given before its meaning?

The distinction is subtle but substantial. HR automates theory growth. Algebraic Abiogenesis aims at theory birth.

3.6 The Ontological Assumptions Retained by HR

HR's power depended on the mathematical structure provided to it. Its primitive concepts, data representations, examples, and production rules were not neutral. They encoded assumptions about what objects existed and what kinds of conceptual transformations were meaningful.

If the system was given integers and divisibility, then number-theoretic ontology was already present. If it was given graphs, vertices, and adjacency, then the basic world of graph theory had already been named. The machine could invent derived concepts, but it did not have to discover that vertices and edges were the appropriate objects.

Its production rules also reflected human mathematical practice. Counting, composing, specializing, generalizing, and quantifying are not arbitrary transformations. They are powerful because mathematicians already know that such operations often produce useful concepts.

This does not reduce HR to a disguised script. Every discovery system requires a language of possible actions. The issue is how much of the eventual theory is already privileged by that language.

The same criticism applies to Algebraic Abiogenesis. Anonymous operations do not eliminate prior commitments. The system still inherits:

- a finite carrier;
- an arity;
- a term grammar;
- a maximum expression depth;
- an enumeration order;
- a method of variable assignment;
- a definition of extensional equality;
- a sampling or enumeration policy for operations;
- complexity measures;
- criteria for retaining or discarding identities;
- rules for comparing emergent theories.

These choices shape what can be discovered. A binary-operation grammar will privilege binary algebra. A shallow term depth may reveal associativity but miss deeper identities. A search over

small carriers may overrepresent locally common laws. Exact term-function equality selects equational regularity while excluding other forms of structure.

The proper contrast is therefore not between a biased HR and an unbiased Algebraic Abiogenesis. No such unbiased system exists. The contrast concerns the level at which interpretation enters.

HR begins with named objects and relations. Algebraic Abiogenesis postpones naming and attempts to derive laws from anonymous operational behavior. Its claim is not absence of ontology, but a thinner initial ontology.

This thinner beginning may permit discoveries that are less constrained by inherited disciplinary categories. It may also produce overwhelming numbers of trivial equivalences. The reduction of prior interpretation therefore increases both openness and difficulty.

HR teaches the central lesson that automated theory formation is possible only when generation, evaluation, proof, counterexample, and concept revision are coordinated. Algebraic Abiogenesis inherits that lesson. Its contribution will depend on demonstrating that anonymous operation-first search produces coherent theories that are not merely rediscoveries forced by the search grammar and not merely finite coincidences without mathematical extension.

4. Conjecture Engines and Data-Guided Mathematics

Between automated theorem proving and full theory formation lies a broad class of systems designed to generate conjectures from mathematical data. These systems do not merely verify statements supplied by humans. They search collections of examples, numerical invariants, symbolic expressions, or structural measurements for regularities that may deserve proof. Their outputs can influence the direction of research by proposing inequalities, identities, classifications, recurrences, or explanatory relationships that were not individually specified in advance.

Conjecture engines represent a substantial advance beyond calculation because they help determine what should be investigated. They transform mathematical data into candidate propositions. Yet they generally operate inside a domain whose objects and descriptive variables have already been chosen. A graph conjecture system may compare graph invariants, but humans have already decided what counts as a graph and which quantities are available for comparison. A formula-discovery system may generate identities for a constant, but the constant, expression grammar, and approximation criterion have already been supplied.

These systems therefore occupy an intermediate position. They can originate mathematical statements, but they usually do not originate the full ontology in which those statements are

expressed. Their history reveals both the power of data-guided discovery and the persistent difficulty of distinguishing meaningful conjectures from the enormous population of statements that are merely true, approximately true, or syntactically novel.

4.1 Graffiti and Automated Inequality Generation

Graffiti was among the most influential systems for automated conjecture generation in graph theory. It was designed to examine numerical properties of graphs and propose inequalities relating those properties. The system worked with graph invariants such as order, size, independence number, diameter, domination number, and other quantities supplied by graph theory.

Its basic discovery mechanism was conceptually simple. Given a database of graphs and their measured invariants, Graffiti searched for numerical relationships that held across all known examples. If one invariant consistently bounded another, the system could formulate a conjectural inequality. The conjecture could then be evaluated by mathematicians, tested against additional graphs, strengthened, weakened, or proved.

The importance of Graffiti was not that it independently created graph theory. It did not discover vertices, edges, adjacency, paths, or graph invariants from anonymous data. Rather, it demonstrated that a computer could participate meaningfully in selecting research questions inside an established field.

Some Graffiti conjectures stimulated published mathematical work. Even false conjectures could be productive when counterexamples revealed previously unnoticed structural distinctions. A proposed inequality might fail only for a narrow family of graphs, thereby motivating a corrected statement or a classification problem.

Graffiti also illustrated the productive role of computational abundance. A human mathematician might consider a small number of plausible relationships among invariants. A machine could systematically compare many quantities over extensive databases. It could detect patterns that would not naturally occur to an individual researcher.

This abundance created an immediate difficulty. The number of possible relationships grew rapidly with the number of invariants and algebraic forms allowed. Most candidate inequalities were weak, redundant, obvious, or special consequences of stronger statements. Graffiti therefore required methods for filtering its own output.

4.2 TxGraffiti and the Filtering of Conjectural Abundance

TxGraffiti extended the conjecture-engine tradition by emphasizing the generation and refinement of graph-theoretic inequalities from computed data. Like its predecessors, it operates over a supplied collection of mathematical objects and invariants. Its central challenge

is not merely finding statements consistent with the data but suppressing conjectures that contribute little new information.

A naive conjecture generator can produce vast numbers of true-looking statements. Suppose a system knows that one invariant is at most ten and another is nonnegative. It can form countless weak inequalities that are technically satisfied by every object in its database. Such conjectures have little mathematical value.

Filtering mechanisms attempt to retain only statements that are comparatively sharp, informative, and nonredundant. One important principle is that a new conjecture should improve on what is already known. If an existing inequality implies the proposed one, the new statement contributes nothing. If the conjecture differs only by a trivial rearrangement, it should not be presented as an independent discovery.

Another principle is extremal relevance. A useful inequality often becomes exact for significant families of examples. Equality cases may reveal structural boundaries and suggest classifications. Conjectures that are never close to equality may be too weak to illuminate the domain.

TxGraffiti and related systems therefore show that conjecture generation cannot be separated from conjecture curation. The machine must maintain a changing background theory against which new candidates are compared. A statement is interesting not only because it fits the data but because it changes the strongest known description of the domain.

This is already a limited form of theory dynamics. Each accepted conjecture alters the standard by which later conjectures are judged. A discovery system must therefore remember previous results, determine implication or domination relationships, and revise its search accordingly.

4.3 The Ramanujan Machine and Formula Discovery

The Ramanujan Machine represents a different form of automated conjecturing. Rather than focusing primarily on inequalities between graph invariants, it searches for formulas involving mathematical constants. Its methods generate candidate continued fractions, series, and related expressions that appear to converge to constants such as π or other fundamental quantities.

The name emphasizes an analogy with Srinivasa Ramanujan, whose notebooks contained many striking formulas presented with little or no proof. The computational system similarly emphasizes the generation of identities and approximations before their complete theoretical explanation is known.

Such work demonstrates that automated systems can search symbolic expression spaces for relationships that are difficult to anticipate manually. A computer can evaluate enormous

numbers of candidate formulas, compare their numerical behavior, identify unusually rapid convergence, and report expressions that appear to encode a deeper identity.

The resulting conjectures can be mathematically valuable even when their proofs remain unavailable. A sufficiently precise numerical agreement across many digits can supply strong evidence, although it does not establish exact truth. The conjecture then becomes a target for analytic proof and theoretical interpretation.

The Ramanujan Machine also exposes the importance of a supplied grammar. The machine does not search every mathematically conceivable statement. It searches particular classes of continued fractions or series determined by human design. The choice of grammar strongly affects what can be found.

This limitation is not unique to machines. Human mathematicians also search through culturally inherited forms. Continued fractions, generating functions, transforms, and special functions are historical vocabularies that make some discoveries visible and others difficult to express.

The relevant question is therefore whether the machine merely instantiates a known formula template or finds parameter combinations and structures that humans did not foresee. In many successful cases, the latter is plausible. Yet the mathematical world in which discovery occurs remains substantially specified in advance.

4.4 Symbolic Regression and the Recovery of Mathematical Laws

Symbolic regression seeks equations that describe observed data while balancing accuracy against expression complexity. Unlike ordinary numerical regression, which estimates parameters inside a fixed model class, symbolic regression searches over the forms of the equations themselves.

Given measurements of variables, a symbolic regression system may combine arithmetic operations, powers, logarithms, exponentials, trigonometric functions, or derivatives to produce an interpretable formula. In scientific applications, such systems can recover known physical relationships or propose compact equations approximating observed behavior.

Symbolic regression is often presented as a mechanism for discovering natural laws from data. Its strength lies in compression. A short equation that accounts for a large body of observations may reveal more structure than a highly flexible numerical predictor.

The method nevertheless depends on substantial prior choices. Researchers determine which variables are measured, which operations are permitted, how errors are scored, how complexity is penalized, and what range of candidate expressions can be explored. A system supplied with positions and times may recover a dynamical law, but it does not necessarily discover that position and time are the correct observational categories.

Noise creates an additional complication. In pure finite mathematics, exact evaluation can establish whether a candidate identity holds over every examined case. In empirical science, approximate fit is unavoidable. A symbolic expression may summarize the data well while failing to represent a true underlying law. Conversely, a genuine law may be obscured by measurement error or hidden variables.

Symbolic regression therefore occupies the boundary between mathematical conjecture and scientific model discovery. It can originate equations, but the semantic interpretation of those equations still depends on human judgment and experimental context.

For Algebraic Abiogenesis, symbolic regression offers both an analogy and a contrast. Both approaches search for compact regularities. Symbolic regression compresses numerical observations into approximate formulas. Algebraic Abiogenesis compresses exact finite behavior into equivalence classes of term functions and candidate identities. The latter sacrifices immediate empirical breadth in exchange for exactness.

4.5 Interestingness, Surprise, and the Selection Problem

The central problem confronting conjecture engines is not generation but selection. Once a system can produce millions of candidate statements, the ability to generate one more is of little value. The decisive task is to determine which conjectures deserve computational, formal, or human attention.

Several possible criteria recur across automated discovery systems.

A conjecture may be considered interesting because it is novel relative to a database of known results. Historical novelty is difficult to establish because the same theorem may appear in different notation, under a different name, or as a consequence of a broader result.

A conjecture may be surprising because its constituent concepts were previously viewed as unrelated. Surprise can sometimes be estimated through statistical rarity or low prior probability, but unusualness alone does not imply importance.

A conjecture may be strong because it implies many weaker statements or sharply constrains the possible behavior of a system. Strength is more formalizable than elegance, although stronger statements may also be less natural.

A conjecture may be compressive because it replaces many separate observations with one law. Compression is a promising measure because mathematics often advances by replacing a collection of facts with a unifying principle.

A conjecture may be productive because it generates new definitions, proof techniques, corollaries, counterexamples, or research questions. Productivity can only be assessed over time or through simulated theory growth.

A conjecture may be explanatory because it reveals why previously disconnected phenomena occur together. Explanation is among the most valued mathematical qualities and among the most difficult to quantify.

No single criterion is sufficient. Maximizing novelty may produce bizarre statements. Maximizing theorem count may favor broad but shallow definitions. Maximizing compression may select trivial summaries. Maximizing proof difficulty may reward technical obstacles without conceptual value.

A mature discovery system therefore requires a portfolio of criteria and a mechanism for resolving conflicts among them. It may also require human judgment at decisive points. The role of the human need not be to inspect every candidate. It may be to establish broad admissibility policies, reject misleading directions, recognize conceptual elegance, and revise the criteria by which the system selects its own output.

The selection problem is especially acute for target-free discovery. When a human supplies a theorem or objective function, success is externally defined. When the system is asked to explore without a fixed target, it must construct a hierarchy of internal goals. This turns interestingness from a secondary ranking mechanism into part of the mathematical ontology itself.

4.6 Why Conjecture Generation Is Not Yet Theory Formation

A conjecture generator produces candidate propositions. A theory-forming system must do substantially more.

A theory contains objects, definitions, relations, laws, examples, counterexamples, proof methods, dependencies, and standards of relevance. Its components are organized rather than merely accumulated. Some concepts are primitive, others derived. Some theorems are central, others routine. Counterexamples define boundaries. Alternative axiom systems may compete. Results may divide the domain into distinct subtheories.

A stream of conjectures does not automatically produce this organization. Even a collection of proved statements may remain mathematically incoherent. A theory requires compression and architecture.

Theory formation also includes vocabulary change. If the existing predicates cannot express the regularity revealed by the data, the system must invent a concept. A conjecture engine restricted to comparisons among supplied variables cannot cross this boundary.

Furthermore, theory formation may require abandoning the original representation. A pattern among numerical invariants may become intelligible only after introducing a structural transformation, equivalence relation, category, or quotient. The decisive advance is then not the conjecture itself but the new ontology in which it becomes natural.

Conjecture systems remain essential components of larger discovery architectures. They supply hypotheses that can be tested, proved, refuted, and reorganized. They demonstrate that machines can contribute to the selection of mathematical questions. Their limitations identify the requirements for the next level.

A transition from conjecture generation to theory formation requires at least five additional capacities: concept invention, proof and countermodel integration, dependency management, dynamic evaluation of interestingness, and the ability to revise the underlying representation.

Algebraic Abiogenesis attempts to move toward this level by treating identities not as isolated outputs but as evidence for emergent equational theories. Exact term-function collisions generate candidate laws. Countermodels test their range. Theory comparison identifies common cores and divergent branches. Recursive search examines whether stable structures persist across operation populations and carrier sizes.

Its success will depend on avoiding the central failure mode of conjecture engines: producing a large catalog of formally valid but conceptually empty statements. The goal is not an archive of collisions. It is the emergence of a mathematical organization that explains why particular collisions belong together, what structures realize them, and what further laws they make possible.

5. Automated Theorem Proving, Countermodels, and Formal Proof

Automated mathematical discovery requires more than generating plausible conjectures. A discovery system must determine whether a proposed law follows from its assumptions, whether it fails in some structure, whether the proof depends on hidden premises, and whether the final result has been expressed with sufficient precision to permit independent verification. These requirements connect machine-generated mathematics to three major computational traditions: automated theorem proving, finite model generation, and interactive formal proof.

Theorem provers search for derivations. Model finders search for structures in which proposed statements fail. Proof assistants verify formal arguments against explicitly specified foundations. Each addresses a different source of mathematical uncertainty. Proof search asks whether a statement can be derived. Countermodel search asks whether it is false under the stated

assumptions. Formal verification asks whether the argument has been encoded without logical gaps, ambiguous notation, or unacknowledged inference.

Together, these tools form a certification ecology. A conjecture may arise from data, symbolic enumeration, or AI-generated speculation. It becomes mathematically credible only after surviving attempts at refutation and, where possible, receiving a proof. For open-ended discovery, however, certification must do more than validate isolated outputs. It must feed back into theory formation. Proofs reveal dependencies, countermodels reveal boundaries, and failed formalizations expose defective definitions. The certification process therefore participates in discovery rather than merely auditing it afterward.

5.1 Resolution, Rewriting, and First-Order Theorem Proving

Automated theorem proving developed from the observation that formal deduction can be represented as systematic symbolic transformation. Given axioms and a conjecture, a prover searches for a sequence of permitted inferences connecting the assumptions to the desired conclusion.

One influential approach is resolution. A problem is translated into a standardized logical form, and the prover repeatedly combines clauses to derive consequences. To prove a statement, the system typically assumes its negation and searches for a contradiction. If contradiction is reached, the original statement follows from the axioms.

Resolution is attractive because it offers a uniform inference mechanism for substantial parts of first-order logic. Its difficulty is combinatorial growth. Every inference can generate additional clauses, most of which are irrelevant to the eventual proof. Effective theorem proving therefore depends on term orderings, clause selection, simplification, subsumption, indexing, and other strategies that reduce the search space.

Rewriting provides another major framework. Equations are oriented into rules that transform expressions toward preferred normal forms. If two expressions reduce to the same normal form, they are equal under the rewrite system. Rewriting is particularly natural in algebra, where identities define transformations among terms.

The orientation of equations is itself mathematically consequential. An equation is symmetric, but a rewrite rule has a direction. Poor choices can cause infinite loops, duplicate work, or prevent convergence. Completion procedures attempt to add new rules so that different reduction paths become consistent. When successful, they can turn an equational theory into an effective decision procedure for equality.

These methods show that proof is not merely a binary certificate. The structure of the proof search reveals which axioms interact, which lemmas are repeatedly useful, and which normal

forms organize the theory. A discovery system can exploit this information. Frequently reused intermediate results may deserve promotion to named lemmas. Recurrent rewrite patterns may indicate a more natural representation. Long proofs may suggest that the axiom system is poorly chosen.

For Algebraic Abiogenesis, automated deduction becomes important after candidate identities emerge from exact evaluation. A term-function collision on one or more finite operations is observational evidence. To determine whether one candidate law follows from others, the system must reason syntactically within the induced equational theory. The distinction between observed law and derivable law is essential. Without it, the system cannot identify minimal axiom sets, redundant identities, or dependency structure.

5.2 Prover9, Vampire, E, and Related Proof-Search Systems

Modern first-order theorem provers implement highly optimized forms of resolution, superposition, rewriting, and equality reasoning. Systems such as Prover9, Vampire, E, and related tools can solve difficult problems by coordinating large numbers of symbolic inferences under carefully designed search strategies.

These provers are often used in benchmark form: a set of axioms and a target theorem are supplied, and the system attempts to produce a derivation within resource limits. Their success has transformed automated reasoning from a philosophical possibility into an established computational discipline.

For mathematical discovery, their value is broader than proving individual targets. A prover can test whether one conjecture follows from a current theory, whether an apparent new law is redundant, whether a proposed axiom can be removed, and whether two formulations are deductively equivalent. It can also expose proof bottlenecks that suggest missing lemmas or defective definitions.

Proof output matters. A prover that reports only success provides less scientific value than one that returns a reconstructible derivation. Proof objects allow independent checking, simplification, translation into proof assistants, and analysis of the reasoning path.

Automated proofs may nevertheless be difficult for humans to interpret. They can contain many low-level steps selected for search efficiency rather than explanatory clarity. A short machine proof is not always conceptually illuminating, while a longer human proof may reveal the central idea more clearly.

This creates a distinction between proof discovery and proof explanation. A machine-generated derivation can establish truth without identifying why the theorem matters or which structural principle makes it natural. A complete AI mathematics system should therefore support proof

compression, lemma extraction, dependency visualization, and translation into human-readable argument.

The use of multiple provers can also improve reliability. Different systems implement different search strategies and may succeed on different problems. Agreement across independent tools reduces the risk that a result depends on one idiosyncratic encoding or implementation.

In an automated theory-forming environment, proof search should be treated as a continual background process. Newly proposed identities can be checked against the current theory. Redundant laws can be removed. Minimal generating subsets can be sought. Unexpected implications can become discoveries in their own right.

5.3 Mace4 and Finite Countermodel Generation

A theorem prover searches for a proof that a statement is universally valid under given assumptions. A model finder searches for a structure satisfying the assumptions while falsifying the proposed conclusion. These are complementary activities.

Mace4 and related finite model finders are especially important in algebra because many false conjectures have small countermodels. A candidate identity may hold in every structure examined so far but fail in a previously untested operation table. A finite countermodel immediately demonstrates that the conjecture is not a theorem of the stated axioms.

Countermodels provide more than rejection. Their structure can reveal exactly why a conjecture fails. A smallest countermodel identifies the minimal environment in which the proposed law breaks. Comparing positive examples and negative countermodels may suggest a missing hypothesis, an overlooked subfamily, or a more informative invariant.

Suppose a candidate identity holds across many anonymous operations but fails in one small table. That failure may indicate that the observed theory divides into two branches. The countermodel may instantiate a neighboring algebraic species governed by a different law. A theory-forming system should preserve that branch rather than treating it as noise.

Finite model search also supports independence analysis. To show that an axiom is not derivable from others, one seeks a model satisfying the remaining axioms but violating the selected one. Repeating this process across an axiom set can identify which laws are independent and which are consequences.

This is crucial when a machine generates many candidate identities. Without independence analysis, the resulting theory may be bloated with redundant statements. A compact basis is mathematically more informative than an undifferentiated catalog of laws.

Finite countermodels have limits. Failure to find one does not prove a statement. The required model may be larger than the search bound, infinite, or difficult to encode. Nevertheless, systematic countermodel search is one of the strongest defenses against overgeneralization from finite evidence.

For Algebraic Abiogenesis, model generation is not merely an external validation step. It is part of the discovery engine. Candidate theories can be tested against newly generated anonymous operations. Operations that satisfy the current law set form one population; operations that violate particular laws define competing populations. Theory growth and countermodel generation therefore become mutually recursive.

5.4 Coq, Agda, Isabelle, HOL, and Lean

Interactive proof assistants differ from conventional automated theorem provers. Rather than relying primarily on unrestricted automatic search, they provide formal languages in which users construct definitions, statements, and proofs that are checked by a small trusted kernel.

Coq, Agda, Isabelle, HOL, Lean, and related systems embody different logical foundations and proof-development styles, but they share a common purpose: to make mathematical arguments explicit enough for mechanical verification.

A proof assistant checks far more than the final conclusion. It checks that every term has the expected type, every definition is well formed, every lemma is invoked under the correct assumptions, and every inference is licensed by the underlying logic. Ambiguities tolerated in ordinary mathematical prose must be resolved.

This precision has major scientific advantages. Formal developments can expose hidden assumptions, circular definitions, missing cases, and invalid appeals to intuition. They also support reusable libraries. Once a theorem is certified, later proofs can build on it without rechecking every informal detail.

The cost has traditionally been substantial human labor. Mathematicians must translate concepts into formal definitions, locate relevant library results, manage coercions and notation, prove routine lemmas, and guide tactics through technical obstacles. The mathematical idea may be short while the formal proof is long.

AI assistance is beginning to reduce this burden. Language models and specialized proof systems can propose tactics, retrieve lemmas, translate natural-language statements, repair errors, and generate portions of formal proofs. Yet semantic correctness remains a central concern. A formally verified theorem may not represent the intended informal claim if the definitions or translation are wrong.

Formalization therefore requires two forms of validation. The proof assistant certifies that the encoded statement follows from the encoded assumptions. Human or automated semantic review must determine that the encoding corresponds to the intended mathematics.

For machine-generated theories, this distinction becomes especially important. The system may invent unfamiliar concepts whose intended meaning is not independently known. Formalization can certify internal coherence, but it cannot by itself establish explanatory value or historical novelty.

5.5 The Computer-Assisted Development of Homotopy Type Theory

The development of Homotopy Type Theory provides an important example of deep interaction between mathematical invention and computer verification. The conceptual program was human-driven, but proof assistants played a substantial role in testing definitions, checking arguments, and developing formal libraries.

Homotopy Type Theory connects type theory with homotopical and higher-dimensional ideas. Its objects, identity structures, and foundational interpretations required unusually precise reasoning about equality, equivalence, and construction. Formal proof systems were therefore not merely convenient calculators. They were natural laboratories for the mathematics itself.

The resulting book and associated research program illustrate a transitional model of computer-assisted mathematics. Human mathematicians supplied the central ontology, foundational ambitions, definitions, and conceptual synthesis. Computers helped verify portions of the emerging structure and supported collaborative development.

This model differs from autonomous theory formation. The machine did not independently identify the need for univalence, invent the interpretation of identity types as paths, or organize the foundational program. Yet it contributed materially to reliability and feasibility.

The episode also demonstrates that mathematical foundations and computational implementation can develop together. A proof assistant is not always an external checker applied after the theory has matured. It can influence which definitions are workable, which constructions are natural, and which gaps become visible.

For future AI-assisted mathematics, this suggests a stronger loop. A human may propose a high-level conceptual direction. AI can formalize candidate definitions, test them in proof assistants, search for consequences, identify inconsistencies, and return the results for judgment. The machine thereby absorbs much of the linear formal labor while the human retains responsibility for conceptual orientation.

This division of labor is especially relevant to polymathic research. A researcher may recognize connections across distant domains but be poorly served by spending months on formal syntax

and library engineering. AI can function as the bridge between nonlinear insight and machine-certified mathematics.

5.6 From Human-Written Proof Scripts to AI-Generated Formalization

The traditional formalization pipeline required humans to write most of the proof script. Automated tactics could solve local goals, but humans selected definitions, decomposed arguments, searched libraries, and repaired failures. Recent AI systems increasingly participate in all of these tasks.

An AI formalizer can begin with an informal theorem, propose a formal statement, generate proof attempts, inspect compiler feedback, retrieve relevant lemmas, revise definitions, and continue until the proof assistant accepts the result. This turns formalization into an iterative machine-guided process rather than a purely manual translation.

The shift is important because formalization has long been a bottleneck. Mathematical literature contains millions of results, only a small fraction of which have been encoded in proof assistants. If AI can automate routine translation and proof construction, formal libraries could grow far more rapidly.

However, proof acceptance must not be confused with successful formalization of the intended theorem. An AI may weaken the statement, introduce an inconsistent assumption, choose an inappropriate definition, or formalize a nearby but different claim. The kernel may correctly verify the wrong mathematics.

Reliable AI formalization therefore requires controls:

- the informal and formal statements should be compared;
- definitions should be inspected for unintended triviality;
- assumptions should be minimized and made explicit;
- generated proofs should be checked by trusted kernels;
- independent tests and countermodels should be used where appropriate;
- provenance should record how the formal artifact was produced;
- human review should focus on semantic alignment rather than low-level syntax.

The strongest future architecture will combine generative AI with exact symbolic systems. The language model supplies flexible translation, planning, and tool orchestration. The theorem prover supplies proof search. The model finder supplies refutation. The proof assistant supplies final certification.

For Algebraic Abiogenesis, such an architecture would support a complete path from finite observation to formal theory. Anonymous operations would generate exact term-function signatures. Collisions would yield candidate identities. Model finders would seek counterexamples. Theorem provers would derive implications and reduce axiom sets. Proof assistants would certify stable results. AI agents would manage the workflow, document the provenance, and translate the output into human-readable mathematics.

The human role would remain decisive. The machine could establish that a family of identities is consistent, independent, and formally productive. It could not automatically guarantee that the resulting theory is conceptually important, genuinely new, or worthy of sustained attention. Those judgments involve mathematical history, explanatory taste, and strategic vision.

The transition from human-written proof scripts to AI-generated formalization is therefore not the elimination of the mathematician. It is a redistribution of labor. The machine increasingly performs the prolonged sequential work of encoding, searching, checking, and repairing. The human increasingly governs direction, interpretation, selection, and meaning.

This transition closes part of the gap between conjecture generation and theory formation. A system can now move from a speculative law to a certified formal object with less manual intervention. The remaining frontier is whether it can also decide which laws should define a theory and which theories deserve to survive.

6. Machine Learning and the New Mathematics of Search

The arrival of machine learning changed the computational study of mathematics by shifting attention from manually designed heuristics toward strategies learned from data, self-play, search histories, and performance feedback. Earlier automated reasoning systems depended heavily on symbolic rules selected by human experts. Modern systems increasingly learn which proof steps, representations, constructions, or programs are likely to succeed. This has greatly expanded the scale of mathematical search.

Machine learning does not remove symbolic structure. The strongest systems usually combine statistical guidance with exact calculation, formal deduction, program execution, or proof verification. A neural model proposes promising directions; a symbolic engine determines whether those directions are valid. This hybrid architecture is especially effective because the two components address different weaknesses. Statistical models navigate enormous spaces but can produce unsupported conclusions. Symbolic systems provide exactness but often search inefficiently without guidance.

The resulting achievements are substantial. Machines have solved difficult geometry problems, produced formal proofs, discovered faster algorithms, improved combinatorial constructions, and guided human conjecture formation. Yet most of these systems operate under a supplied objective. Humans define the theorem, score, benchmark, evaluator, or space of acceptable outputs. The machine discovers a route or construction that performs well under that criterion.

This distinction is central to the present review. Machine learning has transformed mathematical search, but optimized search is not identical to autonomous theory formation. A system may discover a better solution without deciding which problems, concepts, or mathematical worlds should exist.

6.1 Neural Guidance for Symbolic Proof Search

Symbolic theorem proving faces an enormous branching problem. At almost every stage, many inference rules, lemmas, substitutions, or rewrite steps may be applicable. Only a small fraction contribute to a successful proof. Traditional provers manage this problem through carefully engineered heuristics. Neural guidance attempts to learn those choices from prior proofs and failed searches.

A model can be trained to estimate which theorem, lemma, clause, or tactic is most relevant to the current proof state. Rather than explore every possibility uniformly, the prover prioritizes actions assigned higher probability by the learned model. The symbolic engine still checks every step, so the neural component need not be perfectly reliable. It must only improve the ordering of the search.

This division is powerful. Proof corpora contain recurring patterns that are difficult to encode manually. Certain definitions often appear together. Particular lemma families tend to solve characteristic goal shapes. Common substitutions and decompositions can be learned statistically. A model exposed to many formal proofs may recognize these patterns even without possessing a human-style understanding of them.

Neural guidance can also improve through feedback. Successful proof paths become additional training data. Failed searches reveal which choices were unproductive. The system gradually adapts its strategy to the evolving library.

However, the method inherits the biases of the proof corpus. It is strongest where many similar proofs already exist and weaker in unfamiliar foundational settings. A model may prefer conventional proof styles and overlook radically different approaches. It may also exploit superficial correlations that do not generalize beyond the training distribution.

For open-ended discovery, neural proof guidance is therefore an enabling technology rather than a complete solution. It makes proof search more scalable, but it still normally assumes that the language, axioms, and theorem have already been specified.

6.2 AlphaZero-Style Reasoning and Reinforcement Learning

AlphaZero demonstrated that a system could learn complex strategies through self-play without relying on a large corpus of expert games. The broader idea has influenced automated reasoning: a system can explore a formal environment, receive feedback from success or failure, and improve its action policy through reinforcement learning.

In mathematics, the analogue of a game position is a proof state, symbolic expression, program fragment, or partially constructed object. An action may apply a tactic, invoke a lemma, rewrite a term, construct an auxiliary object, or select a branch of search. A completed proof or improved construction supplies a reward.

Reinforcement learning is attractive because it permits a system to learn long-range strategy. A locally unpromising step may be necessary for a later breakthrough. Immediate heuristic scores may fail to recognize such paths, whereas reinforcement learning can assign value based on eventual success.

Self-play has a mathematical analogue in automatically generating problems at the boundary of the system's current ability. One component proposes statements or tasks, while another attempts to solve them. Problems that are too easy provide little learning signal; problems that are impossible produce only failure. Productive training occurs near the moving frontier.

This architecture can create a curriculum without requiring humans to specify every theorem. Yet the surrounding formal environment remains human-designed. The system still needs rules defining valid states, allowable actions, success, and reward. Even self-generated problems occur inside a supplied mathematical universe.

Reward design is particularly dangerous. A system will optimize the measurable objective, which may diverge from mathematical importance. It may favor short proofs of trivial statements, exploit loopholes in the formal encoding, or repeatedly generate tasks that are easy to solve but conceptually empty. Reinforcement learning therefore intensifies the interestingness problem rather than eliminating it.

6.3 AlphaGeometry and Hybrid Symbolic-Neural Deduction

AlphaGeometry exemplifies the hybrid architecture in which neural generation and symbolic deduction cooperate. Geometry problems often require auxiliary constructions that are not explicit in the original statement. A proof may become straightforward only after adding the right point, line, circle, or relation.

A purely symbolic system can verify the consequences of a construction but may struggle to choose among the enormous number of possible additions. A neural model can propose constructions based on patterns learned from many examples. A symbolic engine then derives exact consequences and checks whether the proposed path leads to a proof.

The importance of this approach extends beyond geometry. Many mathematical problems are difficult not because deduction is impossible but because the correct intermediate representation is missing. The decisive act is often the introduction of an auxiliary object.

Hybrid systems can therefore display a limited form of concept invention. They introduce temporary structures that make a proof possible. Yet these inventions occur inside a highly structured geometric language with known predicates, construction rules, and theorem types.

AlphaGeometry illustrates how machine learning can guide symbolic reasoning without replacing it. Neural suggestions increase exploratory reach, while deduction preserves mathematical validity. The resulting proof can be checked independently of the statistical model that proposed it.

For Algebraic Abiogenesis, the analogous challenge is not merely to choose an auxiliary construction inside a known geometry. It is to determine which identities, equivalence classes, and theory partitions should become the persistent conceptual structure of an initially anonymous algebraic environment.

6.4 AlphaProof and Formal Mathematical Competition Solving

AlphaProof and related systems extend learned search into formal theorem-proving environments. Their objective is not to generate plausible natural-language arguments but to construct proofs accepted by a formal verifier.

Competition mathematics provides a demanding benchmark because problems often require ingenuity rather than routine calculation. The statement is supplied, but the proof path may involve unexpected transformations, inequalities, constructions, or decompositions.

Formal proof environments provide a clear success condition. Either the proof term is accepted or it is not. This exact feedback supports reinforcement learning and search at a scale that would be difficult in informal mathematics.

The achievement is significant because it moves machines beyond local tactic completion toward extended proof discovery. The system must coordinate many steps, use libraries, manage intermediate goals, and recover from failed directions.

Nevertheless, competition solving remains target-directed. The problem has been selected, the assumptions are stated, and the desired conclusion is known. The machine is evaluated by whether it reaches that conclusion.

This is a high level of proof discovery but not theory formation. A system may solve a theorem without generating the surrounding concepts that explain why the theorem belongs to a broader mathematical structure. It may also solve many unrelated problems without developing a persistent research program.

The next challenge is to convert successful proof search into cumulative theory growth. Intermediate lemmas should be retained, generalized, connected, and evaluated for reuse. Repeated proof patterns should motivate new abstractions. Without such mechanisms, even extraordinary theorem-solving ability remains episodic.

6.5 FunSearch, AlphaTensor, and AlphaEvolve

FunSearch, AlphaTensor, AlphaEvolve, and related systems demonstrate a different form of machine-generated mathematics: the discovery of programs, algorithms, and constructions that improve a measurable objective.

FunSearch combines language-model generation with an evaluator. The model proposes programs, the evaluator executes and scores them, and successful candidates become the basis for further variation. In this way, the system can search spaces of executable ideas rather than isolated numerical parameters.

AlphaTensor treated matrix multiplication as a search for efficient tensor decompositions. It discovered algorithms using fewer scalar operations for certain matrix sizes. The problem was mathematically rich, but the objective was precise: minimize the number or cost of required operations.

AlphaEvolve generalizes this pattern by using language models to modify algorithms and programs under automated evaluation. It can explore candidate solutions, preserve improvements, combine successful features, and continue evolving them across many iterations.

These systems show that machine-generated programs can embody mathematical discoveries. An improved algorithm is not merely a faster implementation. It may reveal an unexpected decomposition, combinatorial construction, or structural shortcut.

The strength of evaluator-driven discovery is that success is externally checkable. A candidate either attains the score or it does not. Numerical performance, correctness tests, and formal constraints can be measured automatically. This permits enormous search without constant human review.

The limitation is equally clear. The evaluator contains much of the research question. Humans determine what counts as improvement, which outputs are admissible, and which computational environment is relevant. The machine searches effectively, but the purpose of the search is supplied.

6.6 Discovery Under a Supplied Evaluator

A supplied evaluator converts a vague research problem into an optimization problem. Candidate outputs receive scores, and search favors higher-scoring regions. This framework has driven many of the strongest recent results in AI-assisted mathematics.

Evaluators can measure proof completion, formula accuracy, program runtime, number of operations, packing density, combinatorial size, constraint satisfaction, or agreement with data. They can also combine several objectives, such as performance and simplicity.

The evaluator provides clarity and scale. It allows machines to compare candidates without understanding them in a human sense. It also permits automatic iteration, because every result produces feedback.

Yet the evaluator imposes an ontology. It determines which distinctions matter and which are invisible. A system rewarded for minimizing multiplication count may ignore memory use, interpretability, numerical stability, or conceptual elegance. A system rewarded for satisfying many equations may favor trivial identities unless complexity and independence are also considered.

In target-free mathematics, no single external evaluator is available. The system must decide whether a theory is valuable because it is compressive, stable, surprising, general, productive, interpretable, or connected to known mathematics. These criteria can conflict.

Algebraic Abiogenesis therefore requires a family of evaluators rather than one score. Candidate theories may be judged by the number of operations they classify, the compression of term-function behavior, the independence of their laws, their stability across carrier sizes, their resistance to countermodels, their theorem productivity, and their distance from known varieties.

Even this portfolio remains human-designed. The goal is not to eliminate judgment but to make its location explicit. A system should record which discoveries arose from exact behavior and which survived only because of particular ranking policies.

6.7 The Difference Between Optimization and Open-Ended Mathematics

Optimization asks for the best candidate under a specified objective. Open-ended mathematics may require discovering the objectives themselves.

A system searching for a denser packing, faster algorithm, shorter proof, or larger combinatorial construction knows how success will be measured. Its task may be extremely difficult, but the direction is fixed.

A theory-forming system confronts a different problem. It must decide which patterns deserve names, which laws should be primitive, which exceptions define new domains, and which results justify continued exploration. It may need to maintain several incompatible theories because no single ranking determines the winner.

Optimization tends to converge. Open-ended discovery may diversify. One branch may maximize generality, another explanatory simplicity, another computational richness. The mathematical ecosystem grows through differentiation rather than through selection of one global optimum.

This difference does not diminish evaluator-driven achievements. They provide powerful discovery mechanisms and demonstrate that machines can exceed known human constructions. But they should not be mistaken for autonomous mathematical ontology formation.

The most promising future systems will combine the two modes. Open-ended exploration will generate candidate concepts, theories, and questions. Local optimization will then search efficiently within each emerging branch. Proof systems will certify consequences. Countermodels will divide overgeneral theories. Human judgment will revise the governing policies when the machine's internal criteria drift toward triviality.

Machine learning has therefore transformed mathematical search without resolving the deepest problem. It can guide deduction, learn strategy, generate constructions, and optimize executable ideas. The remaining frontier is whether a machine can discover not only better answers, but new mathematical questions and the worlds in which those questions become natural.

7. Agentic Co-Mathematicians and Human-Governed Discovery

The emergence of large language models has introduced a new computational form into mathematics: the agentic co-mathematician. Unlike a conventional theorem prover, which receives a formal goal and searches within a fixed logical system, an agentic system can interpret natural-language instructions, consult literature, write and execute code, invoke symbolic tools, revise hypotheses, organize intermediate results, and communicate its reasoning to human collaborators. It can participate across the research workflow rather than at a single formal stage.

This broader participation creates the possibility of sustained human–AI mathematical collaboration. A human investigator may supply a conceptual direction, recognize an analogy, reject a trivial result, or identify a change of ontology. The AI may then perform the extended linear work required to test the idea: formalizing definitions, enumerating cases, searching for counterexamples, comparing alternatives, maintaining records, and preparing reproducible artifacts.

The resulting system is neither an autonomous mathematician nor a passive instrument. It is a governed research ecology in which different forms of intelligence perform different functions. Human judgment remains central to significance, interpretation, and direction. Machine intelligence contributes endurance, breadth of search, syntactic precision, and computational coordination.

This architecture is especially relevant to open-ended mathematical discovery. When the problem is not merely to prove a supplied theorem but to discover what the theory should contain, research becomes an iterative process involving exploration, criticism, memory, and revision. Agentic systems are beginning to support that process, although their reliability, autonomy, and claims to novelty require careful qualification.

7.1 Large Language Models as Mathematical Research Agents

Large language models differ from earlier symbolic systems because they can operate across natural language, mathematical notation, programming languages, formal proof syntax, and technical documentation. This flexibility allows one model to mediate among forms of representation that were previously handled by separate specialists.

A mathematical research agent may receive a broad instruction such as investigating a pattern, testing a conjectural mechanism, or searching for alternative definitions. It can decompose the task into smaller operations, generate computational experiments, inspect results, revise its approach, and report intermediate conclusions.

This behavior should not be confused with unrestricted mathematical understanding. Language models can produce invalid derivations, invent references, misread computational outputs, and become confident in false claims. Their strength lies less in isolated reliability than in their ability to coordinate external tools and representations.

When connected to exact systems, an agent can use natural-language flexibility without making natural-language plausibility the final standard. It may propose a conjecture, but a symbolic computation checks the examples. It may write a proof sketch, but a theorem prover searches for a derivation. It may formalize the statement, but a proof assistant verifies the result. It may claim generality, but a model finder searches for counterexamples.

The research agent therefore functions most effectively as an orchestrator. It moves information among tools, preserves the purpose of the investigation, and decides which computational action should occur next. The exact systems remain responsible for local certification.

This orchestration role is important because modern mathematical research often requires transitions among many forms of work. A conjecture may begin as verbal intuition, become a numerical experiment, acquire a symbolic formulation, fail under counterexample search, be revised into a theorem, and finally be formalized. An agentic system can maintain continuity across these transitions.

7.2 Tool Use, Computation, Literature Search, and Persistent Memory

A research-grade mathematical agent requires more than conversational ability. It must interact with external tools and maintain a durable record of the investigation.

Computation allows conjectures to be tested across examples. Computer algebra can simplify expressions, calculate invariants, solve equations, and verify identities. Custom programs can enumerate structures, generate random models, classify finite cases, and measure complexity.

Literature search is necessary for novelty assessment. A result that appears new inside the local experiment may already be known under different terminology. The agent must compare candidate discoveries with established theories, databases, preprints, and historical work. This task is difficult because mathematical equivalence may be hidden by notation or disciplinary separation.

Formal tools provide certification. Theorem provers, satisfiability solvers, model finders, and proof assistants can distinguish valid results from persuasive but incorrect reasoning. An agent must know when to invoke each tool and how to interpret its output.

Persistent memory is equally important. Open-ended discovery cannot proceed effectively if each session begins without knowledge of earlier branches, failures, counterexamples, definitions, and decisions. The system must preserve:

- conjectures already tested;
- examples and counterexamples;
- discarded approaches and reasons for rejection;
- proof dependencies;
- alternative definitions;
- computational parameters;

- unresolved questions;
- provenance of generated artifacts.

Without such memory, the agent may repeat failed searches, revive refuted conjectures, or present old observations as new discoveries. With structured memory, the research process becomes cumulative.

A GitHub repository can provide part of this memory architecture. Code, results, issues, branches, commit histories, and documentation create an executable research record. The repository becomes not merely storage but a map of the theory's development.

7.3 Multi-Agent Criticism and Hypothesis Revision

One language model often produces coherent but self-reinforcing errors. Multi-agent systems attempt to reduce this problem by assigning different roles to separate agents or model instances.

One agent may generate conjectures. Another may search for counterexamples. A third may inspect definitions for triviality. A fourth may compare the result with known mathematics. A fifth may attempt formal proof. A coordinating agent may decide whether the hypothesis should be retained, revised, or abandoned.

This division creates productive opposition. A conjecture generator is rewarded for breadth, while a critic is rewarded for finding failure. A proof agent seeks derivability, while a model agent seeks refutation. A novelty agent searches the literature, while an explanatory agent asks whether the result organizes the theory.

The value of multi-agent criticism does not come merely from numerical voting. Several agents can agree on the same false claim because they share training data and reasoning tendencies. The stronger architecture assigns agents different tools, objectives, and evidential burdens.

Hypothesis revision should also be explicit. When a conjecture fails, the system should not simply generate an unrelated replacement. It should analyze the counterexample, identify the violated condition, and determine whether the original claim can be repaired.

Possible revisions include:

- adding a missing hypothesis;
- restricting the relevant class of structures;
- replacing equality with implication;
- weakening a universal claim;

- introducing an invariant that separates positive and negative cases;
- dividing one theory into several branches.

This process resembles scientific theory change. Refutation becomes a source of structure. The machine learns not only that a statement is false but why its conceptual boundary was incorrectly drawn.

For Algebraic Abiogenesis, multi-agent criticism can prevent term-function collisions from being mistaken for universal algebraic laws. One agent may detect identities within a sampled operation. Another may search for operations violating them. A third may test whether the law follows from a smaller basis. A fourth may determine whether the identity corresponds to a known variety. The surviving result is therefore shaped by adversarial examination.

7.4 Human Strategic Intervention in Machine-Led Exploration

Human intervention remains indispensable, but its most valuable form is strategic rather than clerical.

The human need not inspect every generated term, proof step, or operation table. Those tasks are precisely where machines provide the greatest advantage. The human contributes by recognizing when the system is optimizing the wrong criterion, mistaking rediscovery for novelty, overfitting a finite sample, or producing results that are correct but conceptually empty.

Strategic intervention may redirect the entire search. A human may notice that two failed branches share a hidden invariant. A technical obstruction may suggest that the ontology is wrong rather than the proof incomplete. A collection of identities may become intelligible only after changing the unit of analysis from individual laws to theories of operations.

Such interventions are difficult to formalize because they often arise from broad experience, analogy, taste, and historical awareness. They are also precisely the contributions that polymathic researchers may provide most effectively. A nonlinear conceptual leap can reorganize months of linear computation.

The ideal division of labor is therefore asymmetric:

- the human selects broad directions and evaluates significance;
- the agent expands those directions into explicit tasks;
- computational tools execute and test them;
- the agent organizes the results;
- the human judges whether the emerging theory is worth preserving.

This arrangement preserves separation between person and machine. The AI does not need to absorb the mathematician's identity or consciousness. It remains an external research instrument accessed through ordinary interfaces. The important transfer is not of the self but of labor.

Such governance also establishes accountability. The human investigator remains responsible for the claims made, the criteria used, and the interpretation offered. Machine assistance should be documented rather than obscured, particularly when the AI contributes code, proofs, conjectures, or literature synthesis.

7.5 AI-Assisted Concept Recovery and the Rediscovery of Homology

Recent multi-agent experiments have attempted not merely to prove theorems but to recover mathematical concepts from structured data and limited prior knowledge. A notable example is the computational rediscovery of ideas resembling homology from polyhedral or combinatorial information together with linear-algebraic operations.

Such experiments are important because homology is not an isolated proposition. It is a conceptual technology for detecting global structure through algebraic constructions. Recovering it requires more than finding a numerical correlation. The system must introduce intermediate objects and transformations that organize the data.

This moves AI closer to concept invention. The machine participates in constructing a representation through which previously scattered observations become coherent.

Nevertheless, recovery experiments differ from unrestricted discovery. Researchers know the historical concept they hope the system will approach. They select the initial data, operations, and evaluation criteria partly in light of that destination. Success can be measured by similarity to an existing theory.

This does not invalidate the experiment. Rediscovery is a powerful test of whether a system can reconstruct a concept from weaker information. It also provides a controlled environment in which machine behavior can be compared with mathematical history.

The limitation is that the target remains latent in the experimental design. A stronger test would allow the system to produce several competing concepts and evaluate them without assuming that the historically familiar one is necessarily best.

Algebraic Abiogenesis seeks this stronger form. Its objective is not to recover groups, semigroups, lattices, or Boolean algebras on command. Known structures may appear, but the system should also be permitted to form unfamiliar equational families. The evaluation must therefore distinguish fruitful novelty from meaningless deviation.

7.6 FERMAT and Environments for Open-Ended Theory Construction

FERMAT represents an emerging class of systems designed specifically for open-ended mathematical theory construction. Rather than treating mathematics as a sequence of isolated benchmark problems, it models a changing environment of concepts, objects, relationships, and reasoning actions.

Such an environment may represent mathematical knowledge as a graph. Nodes correspond to definitions, examples, conjectures, theorems, counterexamples, or computational procedures. Edges record dependence, implication, analogy, contradiction, or construction.

Agents act on this graph by proposing concepts, testing statements, adding proofs, generating examples, or revising relationships. The theory is therefore a dynamic object rather than a fixed context.

This architecture addresses a weakness of episodic theorem solving. A system that solves one problem after another may not accumulate a coherent mathematical worldview. An open-ended environment encourages persistent theory growth. Intermediate discoveries remain available and can affect later search.

FERMAT also foregrounds mathematical interestingness. If the system is not given a single theorem target, it must decide which regions of the theory graph deserve further attention. Measures may include novelty, connectivity, proof productivity, compression, difficulty, and usefulness to other agents.

The danger is that graph growth can become an end in itself. A large knowledge graph is not necessarily a good theory. It may contain thousands of weakly connected definitions and routine consequences. Open-ended construction therefore requires mechanisms for consolidation, abstraction, pruning, and theory comparison.

Algebraic Abiogenesis can be interpreted as a specialized theory-construction environment focused on finite operations and equational behavior. Its theory graph may include terms, signatures, identities, operation classes, countermodels, implication relations, and emergent varieties. FERMAT offers a broader architectural precedent, while Algebraic Abiogenesis supplies a more austere initial ontology and exact semantic engine.

7.7 The AI Co-Mathematician as a Research-Workflow Architecture

The AI co-mathematician should be understood less as a single model than as a research-workflow architecture. Its defining feature is the coordination of many activities across time.

A mature system would include:

- a natural-language interface for human direction;

- a planner that decomposes research questions;
- a structured memory of prior work;
- computational tools for experiments;
- theorem provers and model finders;
- proof assistants for certification;
- literature-search mechanisms;
- novelty and interestingness evaluators;
- repository management and provenance tracking;
- human checkpoints for strategic judgment.

The value of the architecture lies in continuity. Mathematical research rarely follows a straight line from problem to proof. It involves repeated reformulation, failed attempts, unexpected examples, and shifts in abstraction. The co-mathematician must preserve these transitions and make them available for future reasoning.

Such a system also changes the economics of research. Tasks that previously required prolonged manual programming, formalization, and documentation can be delegated. A single investigator may explore a larger conceptual space without becoming trapped in implementation details.

This does not make the human unnecessary. It makes human attention more selective. The researcher can concentrate on the decisions that cannot yet be reliably automated: identifying important questions, recognizing deep analogies, judging elegance, detecting conceptual fraud, and determining when a computational pattern has become mathematics.

The strongest vision is therefore neither machine replacement nor machine servitude. It is governed complementarity. Human and machine occupy different cognitive roles while remaining clearly separated.

For Algebraic Abiogenesis, the co-mathematician architecture is not an optional convenience. It is part of the experimental method. Anonymous operation spaces can become too large for manual investigation. Candidate theories can proliferate beyond human memory. Proof, countermodel, classification, literature comparison, and archival packaging must be coordinated continuously.

The project's central wager is that artificial intelligence can absorb this linear burden while leaving conceptual sovereignty with the human investigator. If successful, the result will not be an uploaded mathematician or an autonomous mathematical consciousness. It will be a new

form of research partnership in which machines sustain the procedural continuity required for mathematical worlds to emerge.

8. Equational Reasoning and Computational Abstract Algebra

Abstract algebra is especially well suited to machine-assisted discovery because many of its central structures are defined by operations and identities. A group, semigroup, lattice, quasigroup, loop, ring, or Boolean algebra can be described through operations satisfying laws. These laws can be manipulated symbolically, tested on finite structures, compared for implication, and explored through automated proof and countermodel search.

Equational reasoning therefore provides a natural bridge between theorem proving and theory formation. A machine can generate candidate identities, determine which finite operations satisfy them, search for consequences, identify redundant axioms, construct counterexamples, and classify related algebraic systems. Unlike informal mathematical domains that depend heavily on geometric intuition or semantic interpretation, equational theories offer exact computational access.

Yet computational abstract algebra has generally begun with named structures or supplied identities. The machine is asked to study groups, loops, quasigroups, lattices, or magmas under specified laws. Even large-scale projects that enumerate thousands of equations ordinarily begin with a predetermined formal language and a catalog of candidate laws.

Algebraic Abiogenesis attempts to enter one stage earlier. Instead of starting with an equation and asking what it implies, it begins with anonymous finite operations and asks which equations emerge from their behavior. The distinction is between exploring a declared theory and inducing a theory from operational data.

8.1 Universal Algebra as a Natural Domain for Machine Discovery

Universal algebra studies algebraic structures through their operations, identities, homomorphisms, subalgebras, products, and quotients. Rather than treating groups, rings, lattices, and related systems as entirely separate subjects, it examines the general principles shared by classes of structures defined by equations.

This makes universal algebra unusually compatible with computation. A finite algebra can be represented by a finite carrier and one or more operation tables. Terms can be generated recursively. Their values can be calculated exhaustively. Identities can be tested exactly over all assignments of variables.

A machine does not need an intuitive interpretation of an operation table. It can determine whether the operation is associative, commutative, idempotent, medial, distributive, absorptive, or subject to more complex identities simply by evaluation.

Universal algebra also supplies a mature language for comparing theories. One can ask whether one set of identities implies another, whether a class of algebras forms a variety, whether an identity basis is finite, whether axioms are independent, and whether finite models separate competing theories.

These questions can be attacked using theorem provers, rewrite systems, satisfiability solvers, model generators, and exhaustive enumeration. The field therefore offers both a rigorous semantic foundation and a rich computational environment for machine-generated mathematics.

The same properties also expose a challenge. The number of possible terms and identities grows rapidly with depth, variable count, and operation signature. Most generated identities are redundant or trivial. Efficient theory discovery therefore requires compression and selection.

8.2 Terms, Identities, Varieties, and Finite Models

A term is a formal expression constructed from variables and operation symbols. In a binary language, variables such as x, y, and z may be combined into expressions such as x·y, (x·y)·z, or x·(y·z).

When a term is evaluated in a particular algebra, it defines a function from assignments of variables to elements of the carrier. Two syntactically different terms may determine the same function in that algebra. Their equality is then an identity valid in that structure.

An identity is written as an equation between terms and is required to hold under every assignment of its variables. A class of algebras defined by identities forms a variety. Familiar algebraic families can be described this way, including semigroups, groups in an appropriate signature, lattices, and Boolean algebras.

Finite models provide a concrete semantic laboratory. Because the carrier is finite, every assignment can be checked exhaustively. There is no statistical uncertainty. A candidate identity either holds throughout the operation table or it does not.

This exactness is central to Algebraic Abiogenesis. The system does not infer an identity from approximate correlation. It detects that two terms induce precisely the same function across the complete finite domain.

However, validity in one finite algebra does not imply universal validity. A law may be accidental to that operation table or carrier size. To elevate a collision into a broader theory, the system

must compare multiple models, search for countermodels, and determine whether the identity follows from a stable basis.

Finite algebras therefore play two roles. They are sources of candidate laws and instruments for testing their limits.

8.3 Automated Reasoning in Quasigroups, Loops, Magmas, and Lattices

Automated theorem proving has been applied extensively to algebraic systems whose laws can be expressed equationally. Quasigroups, loops, magmas, semigroups, and lattices are especially suitable because their defining properties can often be encoded compactly.

A magma is simply a set equipped with a binary operation. It imposes almost no law beyond closure. This minimality makes magmas a natural testing ground for automated equational reasoning. Additional identities divide the enormous space of magmas into more structured classes.

Quasigroups and loops introduce solvability and identity conditions. Their theories contain large families of related identities, many of which can be investigated computationally. Automated provers can establish implications among laws, while finite model finders can produce structures separating non-equivalent axiom systems.

Lattices offer another rich domain. Their meet and join operations satisfy commutativity, associativity, idempotence, and absorption. Variations such as distributive, modular, and complemented lattices can be characterized through additional identities.

Computational studies in these domains have shown that automated tools can discover short proofs, identify redundant axioms, construct small counterexamples, and reveal unexpected equivalences among identity systems.

Nevertheless, the usual workflow begins with human-selected identities. Researchers ask whether one law implies another, whether a shorter axiom basis exists, or whether two named varieties coincide.

Algebraic Abiogenesis reverses the order. It first observes which terms coincide under anonymous operations and only later asks whether the resulting law set resembles a known class.

8.4 The Equational Theories Project

The Equational Theories Project represents one of the largest recent efforts to map relationships among simple algebraic identities. It investigates short equations in the language of a single binary operation and studies which equations imply which others.

The project combines massive enumeration, automated theorem proving, finite model search, custom algorithms, human mathematical reasoning, and formal verification. Its scale demonstrates that even a minimal algebraic language generates an enormous landscape of distinct theories.

Each equation defines a possible class of magmas. If one equation implies another, the corresponding theory is stronger. The collection of implication relationships forms a large directed structure that can be analyzed computationally.

The project is especially relevant to Algebraic Abiogenesis because both focus on short identities in anonymous binary-operation settings. Both rely on finite models, proof search, countermodels, and exact symbolic reasoning.

The difference lies in how the identities enter the system. The Equational Theories Project begins with a predetermined enumeration of equations. It then determines their logical relationships.

Algebraic Abiogenesis begins with operations and derives candidate equations from term-function equivalence. It therefore treats identities as outputs of semantic behavior rather than as the initial catalog.

The two approaches are complementary. One maps the space of possible laws. The other seeks mechanisms by which particular laws become visible before they are named.

8.5 Enumerated Laws, Implication Graphs, and Counterexamples

When a large collection of identities is available, their relationships can be organized into an implication graph. Each node represents a law or theory, and each directed edge indicates that one law entails another.

Such graphs reveal equivalence classes, minimal axioms, strong and weak regions, and chains of specialization. They also help prevent repeated work. If a new identity is already implied by a known law, it need not be treated as an independent discovery.

Counterexamples define missing edges. To show that one identity does not imply another, a finite algebra satisfying the first but violating the second may be constructed. Small separating models become witnesses to logical distinction.

This structure supports more than classification. It reveals the geometry of theory space. Some identities generate broad families of consequences. Others occupy isolated or weakly connected regions. Some laws collapse into equivalence under surprisingly short proofs.

For machine-generated theories, implication graphs can function as evolving maps. Newly discovered identities can be inserted, compared, merged, or separated. The system can identify

whether a candidate law adds genuine expressive power or merely restates existing information.

Algebraic Abiogenesis can use this machinery to turn raw collision data into theory architecture. Terms that coincide produce candidate equations. Proof search determines implication. Countermodels test independence. Clustering reveals families of operations sharing common law sets.

The resulting theory ecology is more informative than a flat list of identities.

8.6 The Boundary Between Mapping a Theory and Generating One

Mapping a theory assumes that the relevant laws or concepts have already been specified. The task is to determine their consequences, relationships, models, and boundaries.

Generating a theory requires deciding which laws should enter the map in the first place.

This difference may appear small because both activities eventually manipulate equations. Conceptually, however, they are distinct. A complete implication graph over a supplied equation catalog does not explain why those equations were selected or whether another vocabulary would reveal a more natural structure.

Theory generation must also confront representation choice. A family of operations may appear chaotic under one term grammar but simple under another. A useful invariant may require introducing constants, unary operations, higher arities, or quotient constructions. The theory is not merely found inside a fixed language; it may require revision of the language itself.

Algebraic Abiogenesis currently operates within a deliberately restricted setting. It uses finite carriers, anonymous operations, bounded term grammars, and exact evaluation. This restriction makes the search reproducible and mathematically tractable.

Its stronger ambition is to determine whether stable theory structure can emerge without first supplying named laws. If repeated term-function collisions organize operation populations into coherent, generalizable, and deductively productive classes, then the system has done more than map a theory. It has participated in generating one.

The burden of proof is high. Apparent emergence may reflect hidden choices in grammar, enumeration, sampling, or ranking. A law discovered frequently may simply be favored by shallow term depth. A theory may look stable only because the operation population is narrow.

The boundary between mapping and generating must therefore be established experimentally. The system should demonstrate that the induced theories were not explicitly encoded, that alternative operation populations produce different law ecologies, that results survive changes

in search parameters, and that at least some emergent structures possess explanatory or theorem-generating value.

Computational abstract algebra supplies the exact tools needed for this test. Universal algebra provides the language, finite models provide semantics, automated provers provide deduction, and countermodels provide criticism. Algebraic Abiogenesis adds the attempt to make the laws themselves arise from anonymous operational behavior.

9. Algebraic Abiogenesis: Mathematics Before Names and Axioms

Algebraic Abiogenesis is an experimental program for discovering algebraic structure from minimally interpreted finite operations. Its central question is whether a mathematical theory can begin before its objects, axioms, and conventional names have been supplied. Instead of starting with groups, semigroups, lattices, quasigroups, or another recognized class, the system begins with finite carriers and anonymous operation tables. It then constructs formal expressions, evaluates them exactly, detects expressions that induce the same function, and treats repeated equivalences as candidate algebraic laws.

The term abiogenesis is used by analogy with the emergence of organized life from nonliving chemistry. It does not imply creation from literal nothing. The system requires a computational substrate, a finite universe, an operation signature, a grammar of expressions, and policies governing search and selection. The claim is narrower: recognizable algebraic organization may emerge without being explicitly selected as the target.

The experiment therefore reverses the ordinary order of algebraic exposition. Textbooks introduce objects, state axioms, and derive consequences. Algebraic Abiogenesis begins with behavior. Laws are inferred from exact operational regularity. Objects and theories are interpreted only after stable patterns have appeared.

The resulting process is neither conventional theorem proving nor ordinary data fitting. It is an attempt at semantic induction in a finite algebraic environment. The machine asks which expressions are extensionally indistinguishable, which equalities persist across structures, which law sets classify operation populations, and which candidate theories remain stable under counterexample search and further exploration.

9.1 The Operation-First Starting Point

Classical algebra ordinarily begins with a named structure. A semigroup is introduced as a set with an associative binary operation. A group adds an identity element and inverses. A lattice begins with two operations satisfying specified laws. The axioms define what counts as an instance of the theory.

An operation-first approach reverses this relation. The operation is supplied before its interpretation. No initial claim is made that it represents multiplication, conjunction, composition, order, symmetry, or any other familiar concept. It is simply a finite mapping of the form

$A \times A \to A$

for a binary operation on a finite carrier A.

The system then investigates what the operation does. It does not initially ask whether the table belongs to a known class. It asks which formal expressions agree under all assignments of variables, which derived operations recur, and which patterns distinguish this table from others.

This starting point reduces semantic commitment. Associativity is not assumed. Commutativity is not privileged. Identity elements are not named. Even familiar notation may be withheld to avoid suggesting an interpretation.

The method is therefore closer to experimental observation than to axiomatic development. The operation table is a finite world. Expressions are probes. Their evaluations are observations. Repeated equality is evidence of law.

The operation-first starting point also permits theories that do not align neatly with established classifications. A table may satisfy an unusual collection of identities without belonging naturally to a familiar textbook category. Such cases are especially important because they test whether the system can reveal structure that a conventional target-driven search would ignore.

9.2 Finite Carriers and Anonymous Operation Tables

A finite carrier provides the simplest exact environment in which all possible inputs can be examined. If the carrier contains n elements, a binary operation is represented by an $n \times n$ table with entries drawn from the same carrier.

For a labeled carrier, there are $n^{(n^2)}$ possible binary operations. Even for small n, this space grows rapidly. Many tables differ only by relabeling the carrier, while others possess distinct algebraic behavior. Enumeration, sampling, canonicalization, and isomorphism reduction therefore become important computational concerns.

An anonymous table should not be confused with a random table, although random sampling is one possible source. Operations may be exhaustively enumerated for very small carriers, generated under selected constraints, evolved from previous tables, or drawn from structurally diverse populations.

Anonymity means that the system is not told the intended interpretation of the operation. The table may happen to be isomorphic to a familiar operation, but that fact is not supplied in advance. The system must detect its behavioral regularities before any historical label is applied.

Carrier labels themselves are mathematically irrelevant when only the abstract operation matters. Two tables related by a permutation of the carrier represent isomorphic algebras. A robust implementation should therefore distinguish genuine structural variation from mere renaming.

This requirement already introduces a form of quotienting. Raw tables are grouped under isomorphism, while expressions may later be grouped under extensional equality. The discovery process repeatedly replaces surface variation with stable equivalence classes.

Finite carriers are restrictive but scientifically useful. They permit exhaustive evaluation, exact countermodels, reproducible archives, and complete comparison at bounded sizes. They also expose a limitation: identities holding in all examined finite structures may fail in larger or infinite models. The system must therefore avoid equating finite persistence with unrestricted universality.

9.3 Generation of Formal Expressions

The system constructs terms from variables and anonymous operation symbols. For one binary operation $\circ$ and variables x, y, and z, initial expressions include

$$x,\ y,\ z,\ x \circ y,\ y \circ x,\ (x \circ y) \circ z,\ x \circ (y \circ z),$$

together with many increasingly complex nestings.

Term generation may be organized by depth, node count, variable multiplicity, symmetry class, or another complexity measure. The choice matters because the number of expressions grows combinatorially. Without deduplication and pruning, even shallow searches become unmanageable.

A raw syntax tree records how an expression was formed. Two syntactically different trees may later prove behaviorally identical. The grammar should initially preserve these differences because their collapse is precisely the evidence from which identities emerge.

The expression language determines the kinds of laws that can be discovered. A language with only one binary operation can reveal associativity-like, commutativity-like, idempotence-like, medial, absorption-like, and many other equations involving that operation. It cannot directly express constants, inverses, order relations, partiality, probability, or higher-arity structure unless the signature is expanded.

The grammar is therefore not neutral. It constitutes the system's initial observational vocabulary. Algebraic Abiogenesis postpones named concepts, but it still chooses which expressions are permitted to exist.

This limitation should be treated experimentally. Different grammars can be compared. One search may permit repeated variables; another may require linear terms. One may include nullary constants; another may derive candidate constants from unary behavior. One may use one operation; another may investigate interacting operations.

A mature system should be able to record which emergent laws depend on which grammatical commitments. A theory appearing only under one narrow term language may reflect the observer more than the operation.

9.4 Exact Evaluation and Term-Function Signatures

Every term induces a function when interpreted in a particular finite algebra. If a term contains k distinct variables and the carrier has n elements, the term can be evaluated on all n^k assignments.

The ordered list of outputs is the term-function signature. It provides a complete extensional description of the term's behavior in that algebra.

For example, the two associativity terms

$(x \circ y) \circ z$

and

$x \circ (y \circ z)$

have identical signatures exactly when the operation is associative.

Signatures make equality computationally concrete. Instead of manipulating expressions symbolically to guess whether they are equivalent, the system evaluates both everywhere in the finite structure. Equality of signatures is exact, not probabilistic.

The signature also acts as a canonical behavioral representative. Many syntactic terms may map to the same signature. This produces substantial compression. Rather than continue exploring every equivalent expression independently, the system can retain one representative together with the class of expressions it subsumes.

Exact signatures improve search efficiency and provide the semantic foundation for law discovery. They also support theory comparison. The distribution of term functions, the sizes of equivalence classes, and the rate at which new terms collapse onto existing signatures characterize the operation's expressive behavior.

An operation with many collisions may possess strong algebraic regularity. An operation whose terms continue producing new signatures may be behaviorally richer or less constrained. These differences can become measurable features of emergent theory space.

The signature method also introduces a form of certified memory. Once a term function has been computed and hashed, later terms can be compared against it without reevaluating every pair. The system accumulates a persistent behavioral archive rather than repeatedly rediscovering the same equivalences.

9.5 Collisions as Candidate Identities

A collision occurs when two distinct terms produce the same term-function signature. In the observed algebra, the equation

$s = t$

holds under every variable assignment.

Such collisions are the primary raw discoveries of Algebraic Abiogenesis. They convert semantic behavior into symbolic candidate laws.

Not every collision is equally informative. Some arise trivially because the two expressions are syntactic variants generated by redundant rules. Others may follow immediately from shorter identities already discovered. Some hold only in one operation table. Others persist across large populations and carrier sizes.

The system must therefore rank and organize collisions.

A candidate identity may be preferred because it has low syntactic complexity, equates structurally different terms, compresses many other collisions, separates a meaningful class of operations, or supports a large deductive closure.

Minimal representatives are particularly valuable. If a complex identity follows from a shorter law, the shorter law is a better candidate for the theory's basis. The complex identity remains useful as a consequence but should not be mistaken for an independent axiom.

Collision structure may also reveal familiar laws without naming them in advance. Equal signatures for $x \circ y$ and $y \circ x$ reveal commutativity. Equal signatures for $x \circ x$ and x reveal idempotence. Equal signatures for the two association patterns reveal associativity.

The stronger test is whether the system also finds less familiar identities and organizes them coherently. Rediscovery validates the mechanism, but novelty requires emergence beyond the expected catalog.

A collision is therefore an observation, not yet a universal theorem. Its status depends on the scope of the model population and the later deductive analysis.

9.6 Countermodels and the Rejection of Accidental Regularities

Finite exactness prevents error within a given table, but it does not guarantee generality beyond that table. A candidate identity may hold because of an accidental feature of one small operation.

Countermodel search tests the proposed law against other operations. A single operation satisfying the current assumptions while violating the identity proves that the law is not derivable from those assumptions.

In a target-free setting, the assumptions themselves may still be emerging. The system may begin with a population of operations sharing several observed identities. It then asks whether an additional collision persists across the population or whether a countermodel separates a subfamily.

Countermodels therefore perform classificatory work. They divide an apparently unified theory into branches. One branch may satisfy the candidate identity, while another rejects it. The failure reveals that the original population contained multiple algebraic species.

Minimal countermodels are especially informative. They identify the smallest carrier or simplest table at which a proposed generalization fails. Such examples often expose the exact structural feature omitted by the candidate theory.

The system should preserve countermodels as first-class artifacts. They are not merely failed tests. They define theory boundaries, support independence claims, and prevent future agents from repeating the same overgeneralization.

Countermodel generation can also be adversarial. One component of the system proposes broad laws; another actively searches for operations designed to break them. This opposition reduces the chance that frequency or familiarity will be mistaken for necessity.

9.7 Recursive Theory Peeling and Theory Ecologies

A flat list of identities is not yet a theory. Algebraic Abiogenesis therefore requires mechanisms for organizing laws into nested or competing structures.

Recursive theory peeling begins with a population of operations and identifies identities common to all or most members. Those common laws define a provisional core. The population is then partitioned according to additional identities, signature profiles, or countermodel behavior.

Each resulting branch can be analyzed again. New common laws are extracted, exceptional operations are separated, and subtheories emerge recursively.

The resulting structure resembles a phylogeny or ecology rather than a single linear axiom system. Some theories share a common trunk and diverge through additional laws. Others converge semantically despite different syntactic bases. Some branches may be sparse, unstable, or reducible to neighboring classes.

This ecological view is important because open-ended discovery should not assume that one globally optimal theory exists. Different law sets may compress different regions of operation space. One theory may favor broad coverage, another short axioms, another high theorem productivity.

Theory peeling also reduces the danger of premature naming. Instead of immediately labeling a branch as semigroup-like or lattice-like, the system can preserve its internal law structure until historical comparison is performed.

A theory ecology should record:

- the operations realizing each branch;
- the identities common to the branch;
- the identities separating it from neighboring branches;
- implication and independence relations;
- minimal representatives;
- smallest countermodels;
- stability across carrier sizes;
- correspondence with known varieties.

This organization transforms raw computation into an evolving map of algebraic possibility.

9.8 The Emergence of Objects, Laws, and Algebraic Interpretation

The final stage is interpretation. Once stable identities and theory branches have formed, the system or human investigator may recognize familiar algebraic structures.

An anonymous operation satisfying associativity may be interpreted as semigroup-like. Additional identity and inverse behavior may suggest a group. Idempotence, commutativity, and absorption across two operations may indicate lattice structure.

The naming should come after the behavioral evidence, not before it. Historical labels are valuable because they connect the discovery to established mathematics, but they should not silently determine the search.

More difficult cases arise when an emergent theory has no obvious known name. The system must then determine whether the theory is genuinely distinct, a disguised presentation of an existing variety, or a trivial consequence of the search grammar.

This requires canonicalization, literature comparison, automated implication testing, and expert review. Novelty cannot be established by unfamiliar notation alone.

Objects may also emerge at levels beyond the original carrier elements. Term-function equivalence classes can themselves become objects. Theory branches can become nodes in a higher-order space. Morphisms may be defined by homomorphisms, interpretations, refinements, or translations between theories.

Thus the discovery process can generate a hierarchy:

- carrier elements;
- anonymous operations;
- formal terms;
- term functions;
- equivalence classes of terms;
- identities;
- models;
- theories;
- relations among theories.

At each stage, stable equivalence produces a new kind of object. This links Algebraic Abiogenesis to broader questions of object formation, quotient construction, and admissibility-governed identity.

The strongest outcome would not be merely the rediscovery of familiar axioms. It would be the demonstrated emergence of a coherent, nontrivial, reproducible theory that was not supplied as a target, that survives countermodel testing, and that supports further deductions or constructions.

Such a result would not prove that machines can create mathematics from nothing. It would establish something more precise and defensible: under a deliberately sparse observational policy, mathematical laws and theory structure can arise from exact anonymous operational behavior before human naming and axiomatization.

10. Comparative Analysis and the Question of Novelty

The novelty of Algebraic Abiogenesis cannot be established by describing it in isolation. The field already contains decades of work on automated theorem proving, conjecture generation, concept invention, finite model search, open-ended theory construction, and machine-assisted mathematical discovery. Any defensible claim must therefore identify both continuity and difference.

The project belongs unmistakably to the lineage of automated theory formation. It inherits the ambition of AM and HR, the proof–countermodel ecology of automated reasoning, the exact semantics of finite algebra, the search scale of modern machine learning, and the workflow orchestration of agentic co-mathematicians. Its novelty, if sustained experimentally, lies not in any one of these components but in their conjunction under a particularly sparse starting condition.

Algebraic Abiogenesis begins with anonymous finite operations rather than named domains or supplied laws. It generates formal expressions, detects exact term-function equivalence, promotes collisions to candidate identities, tests them against countermodels, organizes surviving laws into theory ecologies, and delays historical interpretation until structure has emerged. The novelty question is therefore not whether machines have previously generated mathematics. They have. The question is whether this particular operation-first route moves automated theory formation beneath conventional algebraic ontology in a manner that produces coherent and nontrivial theories.

10.1 Algebraic Abiogenesis and AM

AM demonstrated that a computer could construct new mathematical concepts by applying heuristic transformations to an existing body of concepts. It explored elementary mathematics, generated definitions, recognized patterns, and proposed conjectures. In historical terms, it established that mathematical discovery could be treated as search through a concept space.

Algebraic Abiogenesis shares AM's refusal to limit computation to proof verification. Both systems generate candidate mathematical structures rather than waiting for every target to be specified by a human.

The primary difference lies in the supplied conceptual substrate. AM began with mathematical concepts and heuristic rules reflecting established forms of mathematical thought. Its search was broad, but it operated inside an already interpreted conceptual environment.

Algebraic Abiogenesis begins with less semantic content. The primitive operation is not declared to be addition, multiplication, conjunction, or composition. Candidate laws arise from exact behavior rather than from applying a library of concept-building heuristics to named mathematical objects.

This does not make Algebraic Abiogenesis free of prior structure. Its term grammar, complexity measure, finite carrier, and evaluation policy remain human choices. The distinction is one of degree and location. AM places substantial mathematical intelligence in the initial concept system. Algebraic Abiogenesis places more of the burden on extensional operational behavior.

10.2 Algebraic Abiogenesis and HR

HR is the most direct historical comparison because it explicitly pursued automated theory formation. It generated concepts, calculated examples, formed conjectures, invoked theorem provers and model generators, and evaluated mathematical interest.

The similarities are substantial. Both systems treat theory as an evolving ecology of concepts, examples, laws, proofs, and counterexamples. Both require mechanisms for filtering abundance and identifying productive structure. Both aim to contribute before the final theorem has been chosen.

The crucial difference is that HR usually begins inside a named mathematical domain. Integers, graphs, groups, or other structures are supplied together with primitive predicates and concept-production rules. HR grows a theory from this initial ontology.

Algebraic Abiogenesis attempts to induce the ontology itself from anonymous operational behavior. It does not begin with the concept of a semigroup or lattice. Such structures may become recognizable only after law clusters have formed.

The strongest historical description is therefore not that Algebraic Abiogenesis replaces HR, but that it moves the HR program downward by one level. HR automates theory development within a domain. Algebraic Abiogenesis attempts to automate the emergence of the domain from finite operations.

This distinction must be demonstrated rather than asserted. If the system merely rediscovers standard laws favored by its grammar, then the difference from HR may be methodological rather than substantive. A stronger result would require unfamiliar but coherent theory branches that survive historical and deductive comparison.

10.3 Algebraic Abiogenesis and Conjecture Generators

Conjecture generators such as Graffiti, TxGraffiti, and the Ramanujan Machine produce mathematical statements not individually supplied by humans. They therefore share with Algebraic Abiogenesis the capacity to originate candidate laws.

The difference lies in the source and role of the conjectures. Conjecture engines usually operate over named objects and predefined observables. Graph invariants, constants, expression templates, or numerical quantities are selected in advance. The machine searches for relationships among them.

In Algebraic Abiogenesis, candidate identities arise from collisions among generated term functions. The system does not begin by selecting associativity, commutativity, or another familiar law as the object of search. The equations are induced from the complete behavior of expressions over anonymous operations.

A second difference concerns organization. Conjecture engines may output ranked statements. Algebraic Abiogenesis aims to organize identities into deductive bases, operation classes, branch structures, and competing theories. The intended product is therefore not a list of conjectures but a theory ecology.

The project will fail to distinguish itself if it produces only large catalogs of equalities. Its contribution depends on showing that the collision process yields higher-order organization: stable law families, explanatory compression, and productive deductive structure.

10.4 Algebraic Abiogenesis and Automated Theorem Provers

Automated theorem provers derive consequences from supplied axioms. They are indispensable to Algebraic Abiogenesis, but they do not perform the same task.

A theorem prover begins after the language, assumptions, and target have been specified. Algebraic Abiogenesis begins earlier, by generating candidate identities from finite semantic behavior. The prover then determines which identities follow from others, whether a proposed basis is redundant, and how the emergent theory is organized deductively.

The distinction can be expressed as one between induction and deduction. Algebraic Abiogenesis induces candidate laws from operation tables. The theorem prover deduces consequences within the resulting law system.

The relationship is recursive. Deductive analysis can alter the inductive process. If many collisions follow from one short identity, that identity gains explanatory importance. If a proposed law contributes no new consequences, it may be removed. Proof structure therefore helps shape the theory that induction first suggested.

Algebraic Abiogenesis should not be presented as an alternative to theorem proving. Its architecture depends on theorem proving to convert finite observations into compressed and certified mathematical structure.

10.5 Algebraic Abiogenesis and FERMAT

FERMAT and related open-ended environments treat mathematical knowledge as a changing network of concepts, conjectures, examples, proofs, and actions. They are contemporary neighbors because they address the problem of persistent theory construction rather than isolated problem solving.

FERMAT is broader in scope. It aims to support open-ended mathematical exploration across symbolic entities and reasoning processes. Algebraic Abiogenesis is narrower and more austere. It focuses on finite operations, bounded term languages, exact signatures, and equational theory.

This narrowness may be an advantage. Exact semantics permit reproducible experiments and sharp novelty claims. Every term-function collision can be reconstructed. Every countermodel can be archived. Every branch of the theory ecology can be tied to explicit finite operations.

FERMAT, however, highlights a challenge that Algebraic Abiogenesis has not fully solved: mathematical interestingness. Exact collisions are abundant. The system must determine which identities deserve promotion, which theory branches deserve resources, and which structures possess explanatory or generative value.

The comparison therefore suggests a productive synthesis. Algebraic Abiogenesis supplies a rigorous test domain for open-ended theory construction, while FERMAT-like architectures supply broader mechanisms for theory memory, interestingness learning, and persistent agent action.

10.6 Algebraic Abiogenesis and the Equational Theories Project

The Equational Theories Project is the nearest large-scale comparison in mathematical subject matter. It studies identities in a single binary operation, their implication relationships, and the finite structures that separate them.

Its achievement is the systematic mapping of an enormous landscape of equational theories. It demonstrates the power of combining automated proof, countermodel search, custom computation, human reasoning, and formal verification.

The decisive difference is the direction of construction. The Equational Theories Project begins with a syntactically enumerated catalog of equations and determines their relationships.

Algebraic Abiogenesis begins with operation tables and derives equations from semantic equivalence among terms.

One maps the logical space of candidate laws. The other attempts to explain how law candidates can emerge before being chosen.

The two projects may eventually converge. Identities induced by Algebraic Abiogenesis can be placed into an implication map of the kind developed by the Equational Theories Project. Conversely, known implication structures can help determine whether an emergent identity is fundamental, redundant, or historically familiar.

Novelty therefore lies not in studying binary identities but in the proposed semantic route by which particular identities enter the theory.

10.7 Algebraic Abiogenesis and Self-Supervised Theorem Discovery

Self-supervised theorem-discovery systems begin with axioms and inference rules, generate proof tasks, solve them, extract useful intermediate results, and build a growing theorem library.

These systems reduce dependence on human-supplied theorem targets. They show that a machine can expand a formal theory through internally generated proof activity.

The starting point remains axiom-first. The language, inference system, and foundational assumptions are supplied. The system discovers theorems inside that formal universe.

Algebraic Abiogenesis is behavior-first. It attempts to induce candidate axioms or law systems from anonymous finite operations before constructing deductive closure.

The distinction is complementary:

anonymous operations → candidate laws → theory basis

versus

axioms → generated proof tasks → theorem library.

A mature autonomous mathematics system might combine both. Algebraic Abiogenesis could generate provisional law systems, while self-supervised theorem discovery could expand each system into a deductive corpus. The productivity of the resulting theorem libraries could then help rank the originating theories.

10.8 The Distinctive Conjunction of Anonymity, Exactness, and Target Freedom

No single feature of Algebraic Abiogenesis is unprecedented.

Anonymous operations are standard in finite algebra. Exact term evaluation is standard in universal algebra. Identity detection, countermodel generation, theorem proving, and theory classification all have established histories. Open-ended theory construction and agentic mathematical research also have precedents.

The potentially distinctive contribution is their conjunction:

- anonymous finite operations as the initial empirical substrate;
- bounded but generative expression languages;
- complete extensional evaluation;
- exact term-function signatures;
- collisions as candidate identities;
- adversarial countermodel search;
- deductive compression into minimal or near-minimal bases;
- recursive partitioning into theory ecologies;
- delayed comparison with named algebraic structures;
- no supplied target theorem or target variety;
- reproducible execution and archival provenance;
- human strategic governance combined with AI linear execution.

An isolated use of any one component would not establish novelty. The research claim concerns the complete pipeline and the type of mathematical output it produces.

10.9 Claims That Can and Cannot Presently Be Defended

Several claims can presently be defended.

Algebraic Abiogenesis is an operation-first architecture for inducing candidate equational theories from anonymous finite operations.

It differs from conventional theorem proving because its candidate laws are outputs rather than inputs.

It extends the historical automated theory-formation program by reducing the amount of named mathematical ontology supplied at the beginning.

It provides an exact, reproducible environment in which questions about machine-generated laws, theory selection, and mathematical object formation can be tested.

It combines methods that are usually studied separately: semantic evaluation, identity induction, countermodel search, theory comparison, and agentic workflow management.

Stronger claims remain unproven.

It cannot yet be claimed that the system creates mathematics from nothing. Substantial representational and evaluative policies remain human-supplied.

It cannot yet be claimed that the system is the first automated theory-forming program. AM, HR, and related systems clearly precede it.

It cannot yet be claimed that it has produced a major new algebraic theory. Novelty requires comparison with existing universal algebra, proof of nontriviality, and evidence of continued theorem productivity.

It cannot yet be claimed that the process is fully autonomous. Human choices govern grammar, operation populations, ranking policies, interpretation, and research direction.

It cannot yet be claimed that term-function collisions alone constitute discovery. They become mathematically significant only when organized, generalized, certified, and shown to illuminate a coherent class of structures.

The most defensible formulation is therefore cautious but substantial:

Algebraic Abiogenesis is an experimental extension of automated theory formation that seeks the emergence of algebraic laws and theory structure from exact behavior of anonymous finite operations before familiar objects, axioms, or target theories are named.

The next burden is empirical. The architecture must demonstrate that it can repeatedly produce stable, nontrivial, historically distinguishable, and deductively productive theory structures under transparent experimental conditions.

11. Evaluation, Certification, and Reproducibility

The success of Algebraic Abiogenesis cannot be measured merely by the number of identities generated, the volume of computation completed, or the unfamiliarity of its outputs. A system can produce enormous quantities of formally correct but mathematically unimportant material. It can also rediscover known laws under obscure notation and mistake unfamiliar presentation for genuine novelty. Evaluation must therefore address several distinct questions. Did the machine generate the result without being given the intended target? Is the result

mathematically correct? Does it survive changes in carrier size, operation population, and search policy? Is it deductively nonredundant? Does it compress or explain a meaningful class of structures? Is it historically new? Can another investigator reproduce the entire discovery process?

These questions require different forms of evidence. Exact evaluation certifies that a collision holds in a particular finite algebra. Countermodels define the limits of generalization. Automated proof establishes implication from an identified law basis. Formal proof assistants certify logical derivations. Literature comparison addresses historical priority. Repeated experiments test robustness. Public repositories preserve provenance and permit independent execution.

No single certificate is sufficient. A formally verified identity may be trivial. A novel finite pattern may not generalize. A robust law may already be classical. A productive theory may depend strongly on a hidden human choice in the search grammar. The evaluation of machine-generated mathematics must therefore be layered, adversarial, and transparent.

11.1 What Counts as a Machine-Discovered Mathematical Theory

A mathematical theory is more than a set of true equations. It contains a population of objects or models, a vocabulary for describing them, a collection of governing laws, and relationships among those laws. It distinguishes primitive assumptions from derived consequences and organizes examples, counterexamples, constructions, and transformations into a coherent structure.

For a machine-generated output to count as a discovered theory, several conditions should be met.

First, the output must exhibit organization. The system should identify a stable core of laws, not merely report isolated collisions. It should determine which identities are fundamental, which are redundant, and which distinguish neighboring classes of operations.

Second, the output must define or isolate a nontrivial model population. A proposed theory should classify more than one accidental operation table unless the single structure itself has exceptional significance. The theory should identify a reproducible family of operations or explain why a particular finite algebra is structurally distinguished.

Third, the theory should possess deductive productivity. Its central laws should generate additional consequences through proof, rewriting, or closure. A law set that supports no meaningful deductions beyond the observations from which it arose remains closer to a data summary than to a mathematical theory.

Fourth, the theory should remain stable under reasonable perturbations of the experiment. Small changes in enumeration order, term representatives, sampling seeds, or implementation details should not destroy its central structure.

Fifth, the output should support interpretation. This need not mean immediate human naming. It means that the system can explain how the theory separates models, compresses behavior, or generates consequences.

Machine discovery does not require complete autonomy. Human investigators may choose the broad domain, approve evaluation criteria, and judge significance. The relevant standard is whether the decisive law structure was explicitly supplied or emerged through the computational process.

11.2 Rediscovery, Novelty, and Historical Comparison

Rediscovery and novelty should be treated as different achievements.

Rediscovery is valuable because it validates the discovery mechanism. If a system beginning with anonymous operations independently recovers associativity, commutativity, idempotence, absorption, mediality, or other recognized laws, then the process has demonstrated that familiar algebraic structure can emerge from exact behavior without being named in advance.

Such recovery is strongest when the system also reconstructs relationships among the laws. It should not merely detect associativity in one table. It should recognize a family of associative operations, identify consequences of associativity, and distinguish associative from nearby nonassociative branches.

Historical novelty requires a higher evidential burden. A law may be absent from a local database but well known under another notation. Two equations may look different while defining equivalent varieties. A theory may duplicate an established class described in another branch of universal algebra.

Novelty assessment should therefore compare candidate results at several levels: literal equation search, variable renaming, duality, term substitution, definitional equivalence, interpretability, and deductive equivalence. Automated literature search can assist, but expert review remains necessary because mathematical concepts often migrate across terminology and disciplines.

The system should preserve three labels rather than force a premature binary classification: known, apparently novel, and unresolved. An apparently novel result becomes a defensible discovery only after sustained comparison and independent scrutiny.

Rediscovery should not be concealed or treated as failure. A transparent system should report exactly which outputs recover known theories and which resist identification. The mixture itself provides evidence about the search process.

11.3 Syntactic Novelty Versus Semantic Novelty

Syntactic novelty is easy to generate. A machine can produce an unfamiliar equation by increasing term depth, permuting variables, or composing known laws into elaborate forms. Such an expression may be absent from the literature while containing no new mathematics.

Semantic novelty concerns the behavior or theory defined by the expression. Two syntactically different identities may have the same consequences. One may be derivable immediately from another. A complicated law may define a familiar variety under a disguised presentation.

The evaluation process must therefore quotient candidate identities by appropriate notions of equivalence. At the simplest level, equations should be normalized under variable renaming, symmetry, and reversal. More deeply, automated provers should test mutual implication between candidate laws or law sets.

If theory T_1 proves every axiom of T_2 and T_2 proves every axiom of T_1, the two presentations are deductively equivalent even if their surface forms differ. Such equivalence does not make the alternative basis worthless. A shorter, more elegant, or computationally effective basis may still be a contribution. But the novelty claim should concern the presentation rather than the underlying theory.

Semantic novelty may also arise from a new model class, a new boundary between known classes, a previously unknown finite basis, or an unexpected relationship among established theories. The discovery need not introduce an entirely unprecedented algebraic universe to be significant.

This distinction protects the project from one of the easiest forms of self-deception: confusing symbolic unfamiliarity with mathematical invention.

11.4 Compression, Explanatory Power, and Theorem Productivity

Compression is one of the strongest available measures of theory quality. A useful law replaces many separate observations with one generative principle. A compact basis may explain thousands of term-function collisions as consequences of a small number of identities.

Compression can be measured at several levels. Syntactic compression compares the size of the retained axiom set with the raw identity catalog. Semantic compression measures how many operation tables or term behaviors are classified by the theory. Computational compression

measures whether the theory permits faster prediction of term values or equivalence than exhaustive evaluation.

Compression alone is not enough. A constant operation may generate extreme collapse among terms, yet the resulting theory may be mathematically simple. A stronger theory may compress more because it has eliminated almost all variation.

Explanatory power requires that the retained laws reveal why the observed regularities occur together. If associativity explains a large family of parenthesization collisions, it provides more than a compact record. It identifies the structural source of those collisions.

Theorem productivity offers another criterion. After a theory basis is proposed, the system can measure how many nontrivial consequences follow, how reusable its lemmas become, and whether new constructions or classifications emerge. A fertile theory should support continued research rather than terminate in a static summary.

These measures must be normalized carefully. A theory that proves everything because its assumptions are inconsistent has maximal apparent productivity and no mathematical value. Consistency checks and model witnesses are therefore essential.

11.5 Robustness Across Finite Operations and Carrier Sizes

A law discovered in one finite algebra may reflect a genuine structural principle or an accident of small size. Robustness testing distinguishes these possibilities.

The system should evaluate candidate identities across diverse operation populations. These may include exhaustive collections at small carrier sizes, random operations, operations evolved to challenge the current theory, and representatives selected to maximize structural diversity.

Persistence across carrier sizes is particularly important. An identity that appears on carriers of size two and three but disappears at size four should not be promoted as a general law without qualification. Conversely, a theory that recurs independently across several sizes gains credibility.

Robustness should also be tested against changes in search policy. The core results should not depend entirely on one variable set, term-depth cutoff, enumeration order, or random seed. Sensitivity analysis should identify which conclusions remain invariant and which are artifacts of the observational protocol.

This does not imply that size-specific theories are uninteresting. Finite algebra contains genuine phenomena restricted to particular cardinalities. The requirement is accurate scope. The system should distinguish a universal candidate from a law of a bounded model population.

A robust result is one whose domain of validity is clearly characterized and reproducibly detected.

11.6 Countermodel Resistance and Generalization

Countermodel search is the principal defense against premature generalization.

Every proposed implication should be subjected to an adversarial search for a structure satisfying the premises and violating the conclusion. The search should begin with small carriers and expand as resources permit. Constraint solvers and finite model generators can target candidate counterexamples more efficiently than undirected random sampling.

Survival against bounded countermodel search does not prove universal validity. It strengthens the conjecture and establishes a verified lower bound on the size of any counterexample. Proof remains necessary for unrestricted generality.

Countermodels should also guide theory repair. If a candidate law fails, the system should compare the countermodel with positive examples and search for the smallest additional condition that restores the pattern. The result may be a refined theorem or a division of the model population into separate branches.

Generalization should proceed through alternating induction and criticism:

observed collision → candidate law → countermodel search → theory refinement → deductive proof.

This cycle is stronger than frequency-based learning because each failure changes the theory's architecture. The machine does not merely lower confidence in a statement. It acquires a new structural witness.

11.7 Formal Certification and Proof-Assistant Integration

Formal certification should be reserved for results that have survived earlier filters. It would be wasteful to formalize every raw collision. The strongest candidates should first be normalized, compared for redundancy, tested across models, and organized into provisional theories.

A proof assistant can then certify several kinds of claims: that an operation table satisfies a stated identity, that one law follows from others, that an axiom basis is sufficient for a theorem, or that a computational procedure correctly enumerates a bounded space.

Formalization does not by itself certify novelty, importance, or intended meaning. It establishes correctness relative to the encoded definitions and assumptions. Semantic review remains necessary to ensure that the formal statement corresponds to the mathematical claim described in prose.

The ideal architecture separates trusted and untrusted components. AI agents may generate definitions, proofs, and code. The final derivations should be checked by a small trusted kernel. Computational certificates should be independently replayable. Operation tables and countermodels should be included as explicit data rather than summarized only in narrative form.

Formal certification is especially valuable when theories are unfamiliar. Human intuition provides less protection against subtle error in a newly generated ontology. Exact checking becomes a substitute for familiarity.

11.8 GitHub Repositories as Executable Scientific Archives

A conventional paper records conclusions and selected methods. An executable repository can preserve the entire discovery process.

For Algebraic Abiogenesis, the repository should contain the term grammar, operation generators, exact evaluation code, canonicalization procedures, raw or compressed signatures, candidate identities, countermodels, proof artifacts, configuration files, and scripts needed to regenerate the principal results.

Version history records how the theory changed. Failed branches and rejected conjectures can be preserved without occupying the main paper. Issues can document unresolved problems. Releases can define stable experimental milestones.

The repository therefore functions as more than supplementary material. It becomes part of the scientific claim. A reader should be able to determine whether the reported theory truly emerged from the stated inputs and policies.

Reproducibility also requires environmental control. Software versions, dependency locks, random seeds, carrier enumerations, and computational limits should be recorded. Expected outputs should be accompanied by checksums or machine-verifiable summaries.

An executable archive also permits continuation. Other investigators can extend carrier sizes, modify grammars, introduce additional operations, or replace ranking policies. The research becomes a platform rather than a closed report.

This continuation function is especially important for machine-generated mathematics. The repository preserves not only what was discovered but how further discovery can proceed.

11.9 Human Oversight, Provenance, and Intellectual Accountability

Human oversight remains necessary at every stage where mathematical meaning, historical novelty, or scientific significance is judged.

The human investigator should not be required to perform every linear computation. That would defeat one purpose of AI-assisted research. Oversight should concentrate on the policies governing the search, the interpretation of results, and the claims made publicly.

Provenance must identify which components were human-specified and which were machine-generated. This includes initial grammars, operation populations, ranking criteria, code, conjectures, proofs, and narrative interpretation. Such disclosure makes the division of labor inspectable.

Intellectual accountability cannot be delegated to an AI system. The named authors remain responsible for verifying that evidence supports the conclusions, that comparisons with prior work are fair, and that uncertainty is stated honestly.

The most dangerous failure is not a visible computational error. It is an inflated conceptual claim built on technically correct output. A system may find a real identity and still fail to discover a theory. It may form a coherent theory and still fail to establish novelty. It may produce a novel finite phenomenon and still fail to justify broad philosophical conclusions about autonomous mathematics.

Human governance should therefore impose graduated claims. The language of observation, conjecture, theorem, theory, and novelty should be used distinctly. Each level should require its own certificate.

The result is not diminished by caution. On the contrary, a transparent hierarchy of evidence makes machine-generated mathematics credible. Algebraic Abiogenesis will be judged not only by what it discovers but by whether it establishes a rigorous standard for showing that a mathematical theory has genuinely emerged.

12. Toward an Autonomous Ecology of Mathematical Discovery

The long-term significance of artificial intelligence for mathematics will not be determined solely by whether machines solve harder benchmark problems. The deeper transformation will occur if computational systems begin to sustain mathematical research programs: generating questions, inventing concepts, testing conjectures, preserving counterexamples, comparing theories, and revising their own representations over extended periods. Such systems would not merely produce answers. They would maintain ecologies of possible mathematics.

Algebraic Abiogenesis provides a restricted experimental setting in which this possibility can be studied. Anonymous finite operations offer exact semantic environments. Generated terms provide a controlled expressive language. Term-function collisions supply candidate laws. Proof search, countermodels, and theory comparison convert those observations into evolving

structures. Human judgment governs the direction and significance of the work, while AI performs much of the sequential labor.

The objective is not to construct a conscious artificial mathematician or transfer a human mind into a machine. A keyboard and monitor provide sufficient separation. The machine may remain an external instrument while assuming increasingly sophisticated responsibilities. What matters is whether the combined system can preserve conceptual continuity, execute long investigations, and produce mathematical organization that neither participant would have produced alone.

The frontier is therefore not machine autonomy in the absolute sense. It is governed autonomy inside a transparent research ecology.

12.1 From Theorem Proving to Ontology Formation

The progression from calculation to theorem proving, conjecture generation, concept invention, and theory formation represents an increasing transfer of mathematical initiative to machines.

A theorem prover operates inside an ontology. The objects, predicates, operations, axioms, and desired conclusion are already present. Its task is to discover a valid path through that world.

Ontology formation occurs when the world itself becomes part of the search. The system must determine which distinctions deserve preservation, which equivalences define stable objects, which operations are fundamental, and which laws organize the resulting domain.

This transition changes the nature of evaluation. A theorem prover can be judged by correctness, proof length, or success rate. An ontology-forming system must also be judged by coherence, compression, fertility, robustness, and explanatory power. It must produce not only valid propositions but useful mathematical worlds.

Algebraic Abiogenesis approaches ontology formation by withholding conventional interpretations from finite operations. The system does not begin by declaring that an operation is multiplication or conjunction. It allows term behavior to determine which equations become visible and which operation populations belong together.

The resulting ontology is still constrained. The system begins with finite sets, total operations, variables, and a term grammar. It therefore does not discover the very idea of algebra from an unrestricted universe. It explores how much algebraic organization can emerge once a thin operational substrate has been supplied.

This limitation is scientifically useful. Ontology formation should be studied in stages rather than treated as an all-or-nothing mystery. A controlled experiment can reveal which structures

emerge under particular observational policies and which require additional representational resources.

The next stage may allow the system to revise its own signature. Repeated behavior could motivate the introduction of constants, unary operations, derived relations, partial operations, or multiple interacting operations. At that point, the language of the theory would no longer remain completely fixed during discovery.

A still more advanced system could compare alternative ontologies. One representation might organize the data through equations, another through order, another through transformations or categories. The system would then search not only for laws inside a language but for the language that best exposes the structure.

12.2 Human Nonlinear Insight and AI Linear Execution

Human and machine intelligence need not imitate one another to collaborate effectively. Their differences can be treated as complementary resources.

Human mathematical insight is often nonlinear. A researcher may move abruptly between domains, notice an analogy without being able to justify it immediately, or sense that a technically correct approach is conceptually wrong. Ideas may emerge through accumulated experience rather than through a visible chain of deductions.

Such thinking can appear scattered when judged by the standards of programming or formal proof. Yet the apparent disorder may be the mechanism by which distant structures are brought into contact. The polymath's value lies partly in the ability to maintain several incomplete conceptual worlds and recognize an unexpected bridge among them.

Programming demands a different mode. Definitions must be explicit. Interfaces must be fixed. Edge cases must be handled. Each step must be translated into a sequence executable by a machine. This discipline is necessary for reproducibility, but prolonged immersion in it can consume the attention required for broad conceptual synthesis.

AI permits a new division of labor. The human can propose the leap. The machine can construct the bridge.

In practice, the human may state a broad objective, recognize a failed ontology, or reject a result as trivial. The AI can translate that intervention into code changes, experiments, exhaustive searches, proof obligations, documentation, and repository updates.

This division is not a rejection of rigor. It can increase rigor by allowing the human to remain at the appropriate level of abstraction while the machine performs exact linear work. The danger

is not that the machine handles implementation. The danger is that the human accepts the machine's implementation without adequate semantic inspection.

The research loop must therefore alternate between modes:

human conceptual intervention → AI operationalization → exact computation → AI organization → human judgment.

Neither participant alone defines the complete process. The human without computational execution may remain at the level of speculation. The machine without conceptual governance may optimize sterile objectives or generate vast quantities of trivial structure.

12.3 The Preservation of Separation Between Mathematician and Machine

The increasing power of AI does not require psychological, biological, or legal fusion between person and machine. Separation can be a design principle.

The keyboard and monitor create a clear boundary. The human issues instructions, observes results, rejects errors, and chooses which branches to continue. The machine remains external, inspectable, and interruptible.

This boundary preserves responsibility. The researcher cannot attribute a false public claim to an autonomous system while still accepting credit for successful results. The human author remains accountable for the interpretation and publication of machine-assisted work.

Separation also protects intellectual independence. An AI system may propose persuasive but misleading formulations. The physical and conceptual distance of an external interface reminds the user that machine output is an artifact for evaluation rather than an internal intuition.

The alternative vision of consciousness transfer is unnecessary for mathematical collaboration. A person's memories, preferences, or research methods need not be uploaded to gain the advantages of machine execution. Structured instructions, repositories, notebooks, and recorded decisions can provide sufficient continuity.

The aim is not to reproduce the mathematician inside software. It is to externalize selected cognitive burdens.

This distinction also clarifies what can persist after the original investigator stops directing the work. A repository may preserve a discovery method, evaluation policy, and open problem list. Future humans and machines can continue the program without claiming that the originating person has survived as a conscious entity.

The preserved object is a research lineage, not necessarily a person.

12.4 Open-Ended Theory Populations and Evolutionary Search

Open-ended mathematical discovery may be better represented as a population of theories than as a single path toward one optimum.

Each candidate theory can be defined by a model population, a law basis, a deductive closure, and a record of counterexamples. Theories may share common cores while differing through additional identities. Some may merge when their apparent differences prove superficial. Others may divide when new countermodels expose hidden heterogeneity.

Evolutionary search offers one mechanism for exploring this landscape. Candidate theories can vary through the addition, removal, substitution, or recombination of identities. Their model populations can be regenerated or challenged. Their productivity can be measured through the consequences and constructions they enable.

Selection need not produce one winner. Different theories may survive because they optimize different criteria. One may offer the shortest basis. Another may classify the broadest model family. Another may support unusually rich derived operations. A fourth may connect unexpectedly to a known domain.

Diversity is therefore not merely computational waste. It protects the system from premature convergence. A single global score might eliminate a theory that later becomes important under a different interpretation.

An ecological architecture should preserve rare but promising branches while allocating most resources to productive regions. It should record why branches were pruned and permit their restoration if later discoveries change the evaluation.

Theory evolution must also remain mathematically disciplined. Random mutation of equations can generate endless novelty without significance. Variation should be connected to observed countermodels, proof dependencies, or structural gaps. A new law should enter because it explains behavior, repairs a failure, or opens a productive branch.

The resulting process resembles neither ordinary theorem proving nor unrestricted symbolic generation. It is a controlled evolution of mathematical organizations under exact criticism.

12.5 Machine-Learned Mathematical Interestingness

Interestingness is the central unsolved problem of open-ended mathematical discovery. Once machines can generate more candidate mathematics than humans can inspect, selection becomes the primary bottleneck.

Fixed measures capture only fragments of value. Short identities are easy to understand but may be trivial. Rare identities may be novel but arbitrary. Strong theories may collapse model diversity. Productive theories may generate many unimportant consequences.

Machine-learned interestingness would infer patterns from the history of mathematical development and from the internal success of the discovery process. The system could learn which concepts later become widely reused, which conjectures create new branches, and which definitions compress previously disconnected results.

Such learning must be handled cautiously. Historical mathematics reflects human institutions, fashions, and available technologies. A model trained to imitate historical attention may reproduce disciplinary bias and overlook forms of mathematics that humans have not valued.

Internal measures can provide a partial alternative. A candidate theory may gain value when it:

- explains many independent collisions;
- supports short proofs of previously separate facts;
- persists across diverse models;
- produces useful derived constructions;
- separates natural operation populations;
- connects distant branches of the theory ecology;
- remains stable under changes in search policy.

Human feedback can supplement these measures. The mathematician may mark a result as elegant, trivial, surprising, or conceptually important. Over time, the system can learn the investigator's standards without treating them as universal truth.

The evaluation policy should remain explicit and revisable. A theory selected under one notion of interestingness should be distinguishable from a theory forced by the raw data. This separation prevents aesthetic or strategic preferences from being disguised as mathematical necessity.

12.6 Cross-Domain Extension Beyond Finite Algebra

Finite algebra is an ideal initial domain because it permits exhaustive evaluation and exact certification. The broader vision, however, extends beyond anonymous binary operations.

Logical systems provide one direction. Instead of beginning with known connectives, a machine could explore anonymous truth-functional operations and discover stable calculi, equivalence

classes, or circuit structures. NAND-based investigations are especially relevant because complex logical behavior can emerge from one primitive operation.

Combinatorics provides another direction. Anonymous incidence relations, transformations, or local assembly rules could generate candidate invariants and classification theories before familiar objects are named.

Dynamical systems could be approached through finite transition rules. The machine could discover equivalence classes of trajectories, conserved patterns, or emergent coarse-grained objects.

Category-theoretic environments present a more difficult extension. Rather than beginning with elements and operation tables, the system could begin with anonymous objects and composable arrows, searching for stable universal patterns, equivalences, and factorization structures.

Scientific applications introduce approximation and noise. Here exact term-function collision must be replaced or supplemented by statistical equivalence, error bounds, and intervention tests. The risk of false regularity becomes substantially greater.

Cross-domain extension should therefore proceed gradually. The finite algebraic setting can establish the architecture's basic principles: target-free generation, exact comparison, countermodel criticism, theory branching, and delayed interpretation. Each additional domain requires a clear account of what replaces exact finite semantics.

The goal is not to impose algebraic methods everywhere. It is to determine whether the deeper discovery pattern generalizes:

minimally interpreted substrate → generated observations → stable equivalences → candidate laws → theory ecology → interpretation.

12.7 Risks of Triviality, Rediscovery, and Hidden Human Targets

The most immediate risk is triviality. Finite operations often produce many term collisions, especially when the carrier is small or the operation is degenerate. A system may generate impressive volumes of output while discovering only constant behavior, projections, or obvious consequences.

A second risk is rediscovery disguised by notation. The machine may produce a familiar identity in a complicated form and present it as new. Deductive comparison and historical search are necessary before novelty is claimed.

A third risk is hidden targeting. Researchers may say that the system is open-ended while choosing grammars, model populations, and scoring functions designed to recover a desired

structure. The target may be absent from the prompt but embedded in the experimental design.

A fourth risk is parameter dependence. A theory may emerge only under one depth cutoff, carrier size, sampling method, or ranking policy. Without sensitivity analysis, accidental artifacts may appear robust.

A fifth risk is anthropomorphic interpretation. The use of words such as discovery, creativity, understanding, and theory can imply mental capacities not demonstrated by the system. Operational definitions should replace rhetorical inflation.

A sixth risk is computational opacity. An agentic system may generate code and results faster than humans can inspect them. Formal checking can certify some claims, but implementation errors, data leakage, and semantic misalignment remain possible.

A seventh risk is publication abundance without consolidation. AI can produce papers, repositories, conjectures, and formal artifacts at extraordinary speed. Without synthesis, the scientific record may become harder rather than easier to understand.

These risks should be treated as research problems. A credible system should include triviality detectors, known-theory matching, target-audit procedures, parameter perturbation, independent reruns, and explicit uncertainty labels.

The standard should not be whether the machine produces something unfamiliar. It should be whether the process makes a mathematically disciplined distinction between accident, rediscovery, refinement, and genuine novelty.

12.8 A Research Agenda for Post-Axiomatic Mathematical Discovery

Post-axiomatic discovery does not reject axioms. It investigates how candidate axioms can emerge before they are declared fundamental.

A practical research agenda begins with replication. The operation-first pipeline should be executed across independently generated finite operation populations and carrier sizes. The same stable theory branches should recur under transparent conditions.

The next requirement is deductive compression. Large collision sets should be reduced to small bases, with explicit proofs that the remaining identities follow.

Countermodel search should then establish boundaries and independence. Each proposed implication should be challenged, and minimal separating structures should be preserved.

Historical comparison must determine which branches correspond to known varieties and which remain unidentified. This comparison should include alternative presentations and not rely on literal equation matching alone.

The system should then measure theorem productivity. Each theory basis can seed self-supervised theorem generation, derived-operation search, construction discovery, and classification experiments.

A higher-order theory space should be built in which branches are related by implication, interpretation, refinement, duality, and shared model populations.

The research program should also test grammar revision. Can recurrent behavior motivate the introduction of constants, secondary operations, or quotient objects without direct human specification?

Formal certification should accompany the strongest claims. Proof assistants can verify implication chains, computational enumeration procedures, and bounded completeness results.

Finally, the system should be evaluated by independent teams. Reproduction by the original developers is necessary but insufficient. Outside investigators should be able to regenerate the results, modify the policies, and determine whether the reported theory ecology persists.

This agenda converts a philosophical ambition into a sequence of falsifiable experiments.

12.9 Can Mathematics Be Born Before It Is Named?

Mathematics cannot arise from literal nothing. Every discovery process begins with distinctions, representations, and permissible operations. Even a human mathematician inherits language, notation, examples, and standards of proof.

The meaningful question is whether names and axioms must come first.

Algebraic Abiogenesis proposes that behavior can precede declaration. Anonymous operations can be examined before they are classified. Terms can be generated before laws are selected. Exact collisions can reveal regularities before a theory has been named. Countermodels can divide the emerging world before its taxonomy is known.

If stable law systems then organize model populations, support deductions, survive perturbation, and resist historical reduction, mathematics has emerged in a significant operational sense. The system has not created structure from nothing. It has converted minimally interpreted behavior into a new level of formal organization.

The strongest result would be a theory whose discovery path is fully reproducible but whose final structure was not anticipated by the human investigators. It would possess explicit models,

a compact law basis, nontrivial consequences, clear boundaries, and a defensible relation to prior mathematics.

Such an achievement would not eliminate human mathematics. Human judgment would remain necessary to recognize meaning, compare histories, choose future directions, and decide whether the theory deserves attention.

The machine would have supplied something different: the sustained linear process through which a mathematical world became visible.

The answer to the title question must therefore remain conditional. Mathematics may be born before it is named if birth is understood as the emergence of stable, generative, and certifiable structure from behavior. Naming follows when humans recognize the structure, interpret its significance, and place it within the larger history of mathematical thought.

13. References

1. Newell, A., and Simon, H. A. “The Logic Theory Machine: A Complex Information Processing System.” IRE Transactions on Information Theory 2, no. 3 (1956): 61–79. https://doi.org/10.1109/TIT.1956.1056797
2. Newell, A., Shaw, J. C., and Simon, H. A. “Report on a General Problem-Solving Program.” Proceedings of the International Conference on Information Processing, 1959, 256–264.
3. Wang, H. “Proving Theorems by Pattern Recognition I.” Communications of the ACM 3, no. 4 (1960): 220–234. https://doi.org/10.1145/367177.367224
4. Lenat, D. B. AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search. Stanford Artificial Intelligence Laboratory Memo AIM-286, Stanford University, 1976.
5. Lenat, D. B. “EURISKO: A Program That Learns New Heuristics and Domain Concepts.” Artificial Intelligence 21, nos. 1–2 (1983): 61–98. https://doi.org/10.1016/S0004-3702(83)80005-8
6. Lenat, D. B., and Brown, J. S. “Why AM and EURISKO Appear to Work.” Artificial Intelligence 23, no. 3 (1984): 269–294. https://doi.org/10.1016/0004-3702(84)90016-X
7. Ritchie, G. D., and Hanna, F. K. “AM: A Case Study in AI Methodology.” Artificial Intelligence 23, no. 3 (1984): 249–268. https://doi.org/10.1016/0004-3702(84)90015-8
8. Colton, S., Bundy, A., and Walsh, T. “Automatic Concept Formation in Pure Mathematics.” Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 1999, 786–791.
9. Colton, S., Bundy, A., and Walsh, T. “Automatic Invention of Integer Sequences.” Proceedings of the Seventeenth National Conference on Artificial Intelligence, 2000, 558–563.
10. Colton, S., Bundy, A., and Walsh, T. “On the Notion of Interestingness in Automated Mathematical Discovery.” International Journal of Human-Computer Studies 53, no. 3 (2000): 351–375. https://doi.org/10.1006/ijhc.2000.0394
11. Colton, S. Automated Theory Formation in Pure Mathematics. London: Springer, 2002. https://doi.org/10.1007/978-1-4471-0147-5
12. Colton, S. “Automated Conjecture Making in Number Theory Using HR, Otter and Maple.” Journal of Symbolic Computation 39, nos. 5–6 (2005): 593–615. https://doi.org/10.1016/j.jsc.2005.01.003

13. Fajtlowicz, S. “On Conjectures of Graffiti.” Discrete Mathematics 72, nos. 1–3 (1988): 113–118. https://doi.org/10.1016/0012-365X(88)90199-9

14. Larson, C. E. “A Survey of Research in Automated Mathematical Conjecture-Making.” In Graphs and Discovery, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 69, 297–318. Providence: American Mathematical Society, 2005.

15. Larson, C. E., and Van Cleemput, N. “Automated Conjecturing I: Fajtlowicz’s Dalmatian Heuristic Revisited.” Artificial Intelligence 231 (2016): 17–38. https://doi.org/10.1016/j.artint.2015.10.002

16. Caro, Y., Davila, R., Henning, M. A., and Pepper, R. “Conjectures of TxGraffiti: Independence, Domination, and Matchings.” Australasian Journal of Combinatorics 84, no. 2 (2022): 258–274.

17. Davila, R. “Automated Conjecturing in Mathematics with TxGraffiti.” arXiv, 2024. https://doi.org/10.48550/arXiv.2409.19379

18. Raayoni, G., Gottlieb, S., Manor, Y., Pisha, G., Harris, Y., Mendlovic, U., Haviv, D., Hadad, Y., and Kaminer, I. “Generating Conjectures on Fundamental Constants with the Ramanujan Machine.” Nature 590 (2021): 67–73. https://doi.org/10.1038/s41586-021-03229-4

19. Schmidt, M., and Lipson, H. “Distilling Free-Form Natural Laws from Experimental Data.” Science 324, no. 5923 (2009): 81–85. https://doi.org/10.1126/science.1165893

20. Hillar, C., and Sommer, F. “Comment on the Article ‘Distilling Free-Form Natural Laws from Experimental Data.’” arXiv, 2012. https://doi.org/10.48550/arXiv.1210.7273

21. Birkhoff, G. “On the Structure of Abstract Algebras.” Proceedings of the Cambridge Philosophical Society 31, no. 4 (1935): 433–454. https://doi.org/10.1017/S0305004100013463

22. Burris, S., and Sankappanavar, H. P. A Course in Universal Algebra. New York: Springer, 1981.

23. McKenzie, R. N., McNulty, G. F., and Taylor, W. F. Algebras, Lattices, Varieties. Volume I. Monterey: Wadsworth and Brooks/Cole, 1987.

24. McCune, W. “Mace4 Reference Manual and Guide.” arXiv, 2003. https://doi.org/10.48550/arXiv.cs/0310055

25. McCune, W. Prover9 and Mace4. Argonne National Laboratory and University of New Mexico, 2009. https://www.cs.unm.edu/~mccune/prover9/

26. Schulz, S. “E—A Brainiac Theorem Prover.” AI Communications 15, no. 2–3 (2002): 111–126.

27. Kovács, L., and Voronkov, A. “First-Order Theorem Proving and Vampire.” In Computer Aided Verification, Lecture Notes in Computer Science 8044, 1–35. Berlin: Springer, 2013. https://doi.org/10.1007/978-3-642-39799-8_1

28. Nieuwenhuis, R., and Rubio, A. “Paramodulation-Based Theorem Proving.” In Handbook of Automated Reasoning, edited by A. Robinson and A. Voronkov, 371–443. Amsterdam: Elsevier and MIT Press, 2001.

29. Baader, F., and Nipkow, T. Term Rewriting and All That. Cambridge: Cambridge University Press, 1998. https://doi.org/10.1017/CBO9781139172752

30. Bertot, Y., and Castéran, P. Interactive Theorem Proving and Program Development: Coq’Art—The Calculus of Inductive Constructions. Berlin: Springer, 2004. https://doi.org/10.1007/978-3-662-07964-5

31. Bove, A., Dybjer, P., and Norell, U. “A Brief Overview of Agda—A Functional Language with Dependent Types.” In Theorem Proving in Higher Order Logics, Lecture Notes in Computer Science 5674, 73–78. Berlin: Springer, 2009. https://doi.org/10.1007/978-3-642-03359-9_6

32. Nipkow, T., Paulson, L. C., and Wenzel, M. Isabelle/HOL: A Proof Assistant for Higher-Order Logic. Berlin: Springer, 2002. https://doi.org/10.1007/3-540-45949-9

33. de Moura, L., Kong, S., Avigad, J., van Doorn, F., and von Raumer, J. “The Lean Theorem Prover.” In Automated Deduction—CADE-25, Lecture Notes in Computer Science 9195, 378–388. Cham: Springer, 2015. https://doi.org/10.1007/978-3-319-21401-6_26

34. The Univalent Foundations Program. Homotopy Type Theory: Univalent Foundations of Mathematics. Institute for Advanced Study, 2013. https://doi.org/10.48550/arXiv.1308.0729

35. Pelayo, Á., and Warren, M. A. “Homotopy Type Theory and Voevodsky’s Univalent Foundations.” Bulletin of the American Mathematical Society 51, no. 4 (2014): 597–648. https://doi.org/10.1090/S0273-0979-2014-01456-9

36. Davies, A., Veličković, P., Buesing, L., Blackwell, S., Zheng, D., Tomašev, N., et al. “Advancing Mathematics by Guiding Human Intuition with AI.” Nature 600 (2021): 70–74. https://doi.org/10.1038/s41586-021-04086-x

37. Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., et al. “Discovering Faster Matrix Multiplication Algorithms with Reinforcement Learning.” Nature 610 (2022): 47–53. https://doi.org/10.1038/s41586-022-05172-4

38. Trinh, T. H., Wu, Y., Le, Q. V., He, H., and Luong, T. “Solving Olympiad Geometry without Human Demonstrations.” Nature 625 (2024): 476–482. https://doi.org/10.1038/s41586-023-06747-5

39. Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., et al. “Mathematical Discoveries from Program Search with Large Language Models.” Nature 625 (2024): 468–475. https://doi.org/10.1038/s41586-023-06924-6

40. Hubert, T., et al. “Olympiad-Level Formal Mathematical Reasoning with Reinforcement Learning.” Nature, 2025. https://doi.org/10.1038/s41586-025-09833-y

41. Chervonyi, Y., Trinh, T. H., Olšák, M., Yang, X., Nguyen, H., Menegali, M., et al. “Gold-Medalist Performance in Solving Olympiad Geometry with AlphaGeometry2.” arXiv, 2025. https://doi.org/10.48550/arXiv.2502.03544

42. Novikov, A., Vũ, N., Eisenberger, M., Dupont, E., Huang, P.-S., Wagner, A. Z., et al. “AlphaEvolve: A Coding Agent for Scientific and Algorithmic Discovery.” arXiv, 2025. https://doi.org/10.48550/arXiv.2506.13131

43. Georgiev, B., Gómez-Serrano, J., Tao, T., and Wagner, A. Z. “Mathematical Exploration and Discovery at Scale.” arXiv, 2025. https://doi.org/10.48550/arXiv.2511.02864

44. Tsoukalas, G., Saha, R., Thakur, A., Reguyal, S., and Chaudhuri, S. “Learning Interestingness in Automated Mathematical Theory Formation.” arXiv, 2025. https://doi.org/10.48550/arXiv.2511.14778

45. Bolan, M., Breitner, J., Brox, J., Carlini, N., Carneiro, M., van Doorn, F., et al. “The Equational Theories Project: Advancing Collaborative Mathematical Research at Scale.” arXiv, 2025. https://doi.org/10.48550/arXiv.2512.07087

46. Equational Theories Project. “Mapping the Implication Relations among Equational Laws for Magmas.” GitHub repository and project archive, 2024–2026. https://github.com/teorth/equational_theories

47. Berlioz, L., and Melliès, P.-A. “The Latent Space of Equational Theories.” arXiv, 2026. https://doi.org/10.48550/arXiv.2601.20759

48. Aggarwal, D., Kim, O., Ek, C. H., and Mishra, C. “Discovering Mathematical Concepts through a Multi-Agent System.” arXiv, 2026. https://doi.org/10.48550/arXiv.2603.04528

49. Xia, H., Gomes, C. P., Selman, B., and Szeider, S. “Agentic Neurosymbolic Collaboration for Mathematical Discovery: A Case Study in Combinatorial Design.” arXiv, 2026. https://doi.org/10.48550/arXiv.2603.08322

50. Ota, K., Osa, T., and Harada, T. “Self-Supervised Theorem Discovery in a Formal Axiomatic System.” arXiv, 2026. https://doi.org/10.48550/arXiv.2606.28747

51. Zheng, D., von Glehn, I., Zwols, Y., Beloshapka, I., Buesing, L., Roy, D. M., et al. “AI Co-Mathematician: Accelerating Mathematicians with Agentic AI.” arXiv, 2026. https://doi.org/10.48550/arXiv.2605.06651

52. Feng, T., et al. “Towards Autonomous Mathematics Research.” arXiv, 2026. https://doi.org/10.48550/arXiv.2602.10177

53. Yang, K., Poesia, G., He, J., Li, W., Lauter, K., Chaudhuri, S., and Song, D. “Formal Mathematical Reasoning: A New Frontier in AI.” arXiv, 2024. https://doi.org/10.48550/arXiv.2412.16075

54. Google DeepMind. “Advanced Version of Gemini with Deep Think Officially Achieves Gold-Medal Standard at the International Mathematical Olympiad.” July 21, 2025. https://deepmind.google/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-international-mathematical-olympiad/

55. Sandve, G. K., Nekrutenko, A., Taylor, J., and Hovig, E. “Ten Simple Rules for Reproducible Computational Research.” PLoS Computational Biology 9, no. 10 (2013): e1003285. https://doi.org/10.1371/journal.pcbi.1003285

56. Wilson, G., Aruliah, D. A., Brown, C. T., Chue Hong, N. P., Davis, M., Guy, R. T., et al. “Best Practices for Scientific Computing.” PLoS Biology 12, no. 1 (2014): e1001745. https://doi.org/10.1371/journal.pbio.1001745

57. Peng, R. D. “Reproducible Research in Computational Science.” Science 334, no. 6060 (2011): 1226–1227. https://doi.org/10.1126/science.1213847

58. Youvan, D. C. Mathematical It from Bit: Algebraic Abiogenesis, Anonymous Finite Operations, and the Automated Birth of Abstract Algebra. July 2026. https://doi.org/10.13140/RG.2.2.25987.26403

59. Youvan, D. C. No-Corpus Artificial Intelligence from Algebraic Abiogenesis: Policy-Governed Object Formation, Theory Ecologies, and Certified Memory. July 2026. https://doi.org/10.13140/RG.2.2.12146.06088

The author has published over 5,700 AI-assisted papers at: https://www.researchgate.net/profile/Douglas-Youvan/research . Google Scholar has indexed these preprints, and if you want a particular reference, the AI mode of a Google search (Gemini) has efficiently catalogued these preprints.